Model-based

Verification, Optimization, Synthesis and Performance Evaluation of Real-Time Systems

Kim G. Larsen
Aalborg University, DENMARK

Timed Automata

.. and Prices, Games, Probabilities

Kim G. Larsen
Aalborg University, DENMARK

Aalborg

Jomfru Ane Gade

Aalborg University lea
publ

CISS – Center For Embedded Software Systems

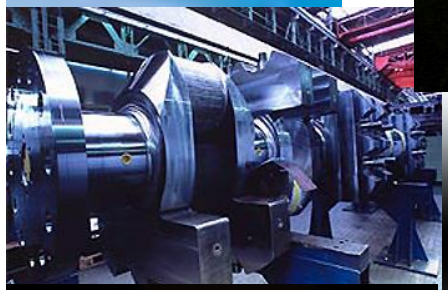
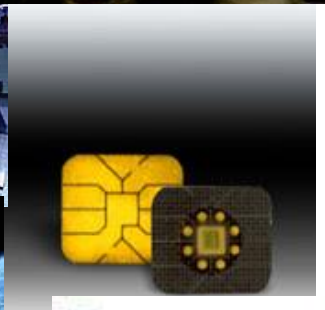
Regional ICT Center (2003-)

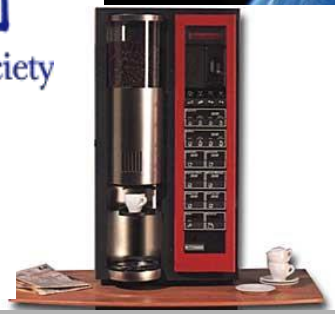
- 3 research groups
  - Computer Science
  - Control Theory
  - HW/SW- codesign

- 20 Employed
- 25 Associated
- 20 PhD Students
- 50 Industrial projects
- 10 Elite-students
- 65 MDKK

- ARTIST Design
- ARTEMIS



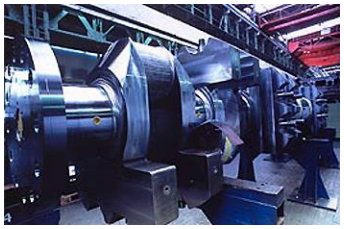



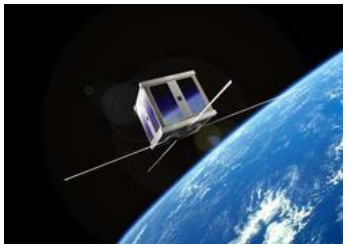



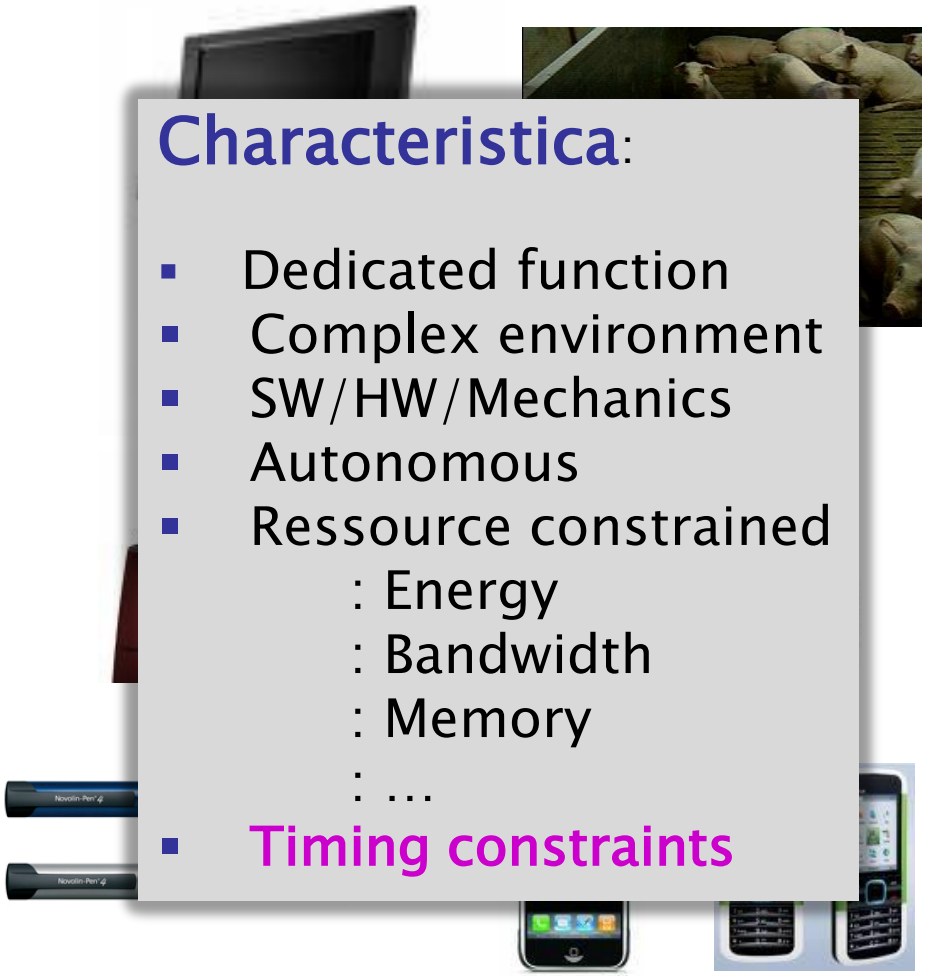
Information Society Technologies

ARTEMIS

ES are Pervasive







Characteristica:

- Dedicated function
- Complex environment
- SW/HW/Mechanics
- Autonomous
- Ressource constrained
  - : Energy
  - : Bandwidth
  - : Memory
  - : …
- Timing constraints

ES are often Safety Critical

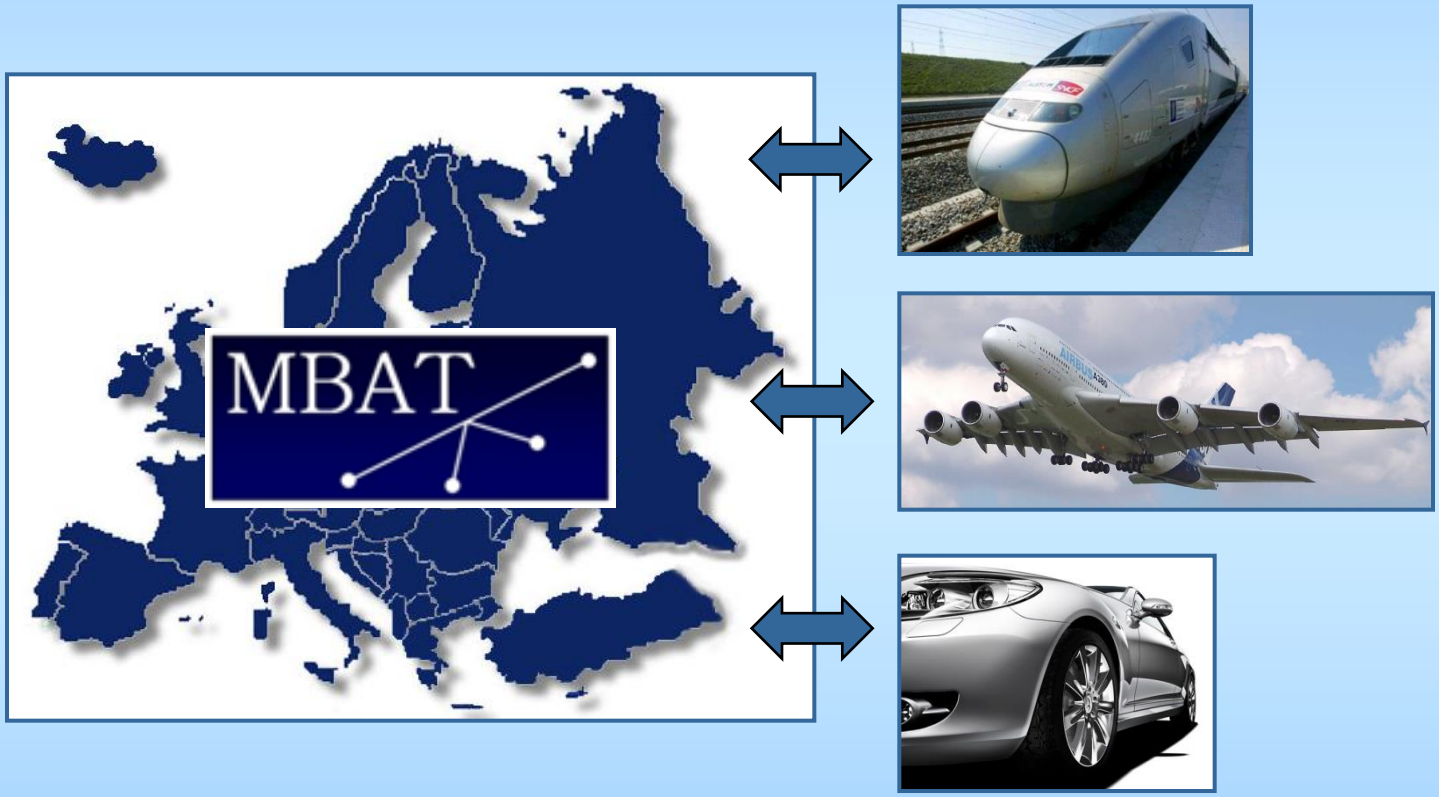
300 horse power
100 processors

How to achieve ES that are:

- correct
- predicable
- dependable
- fault tolerant
- ressource minial
- cheap

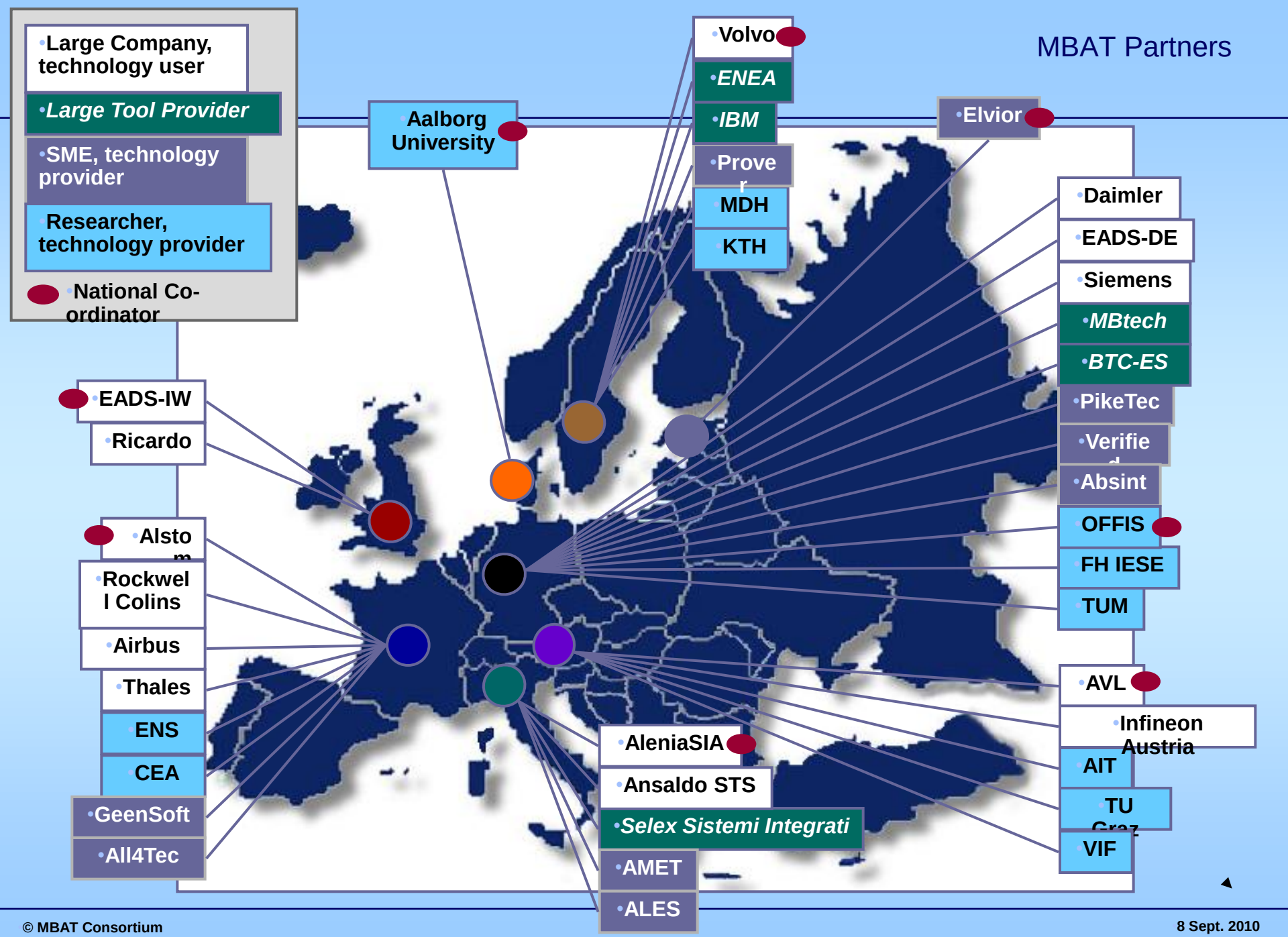
Model-Based Development

- **MBAT will enable the production of high-quality and short-time-to-market transportation products at reduced development costs**



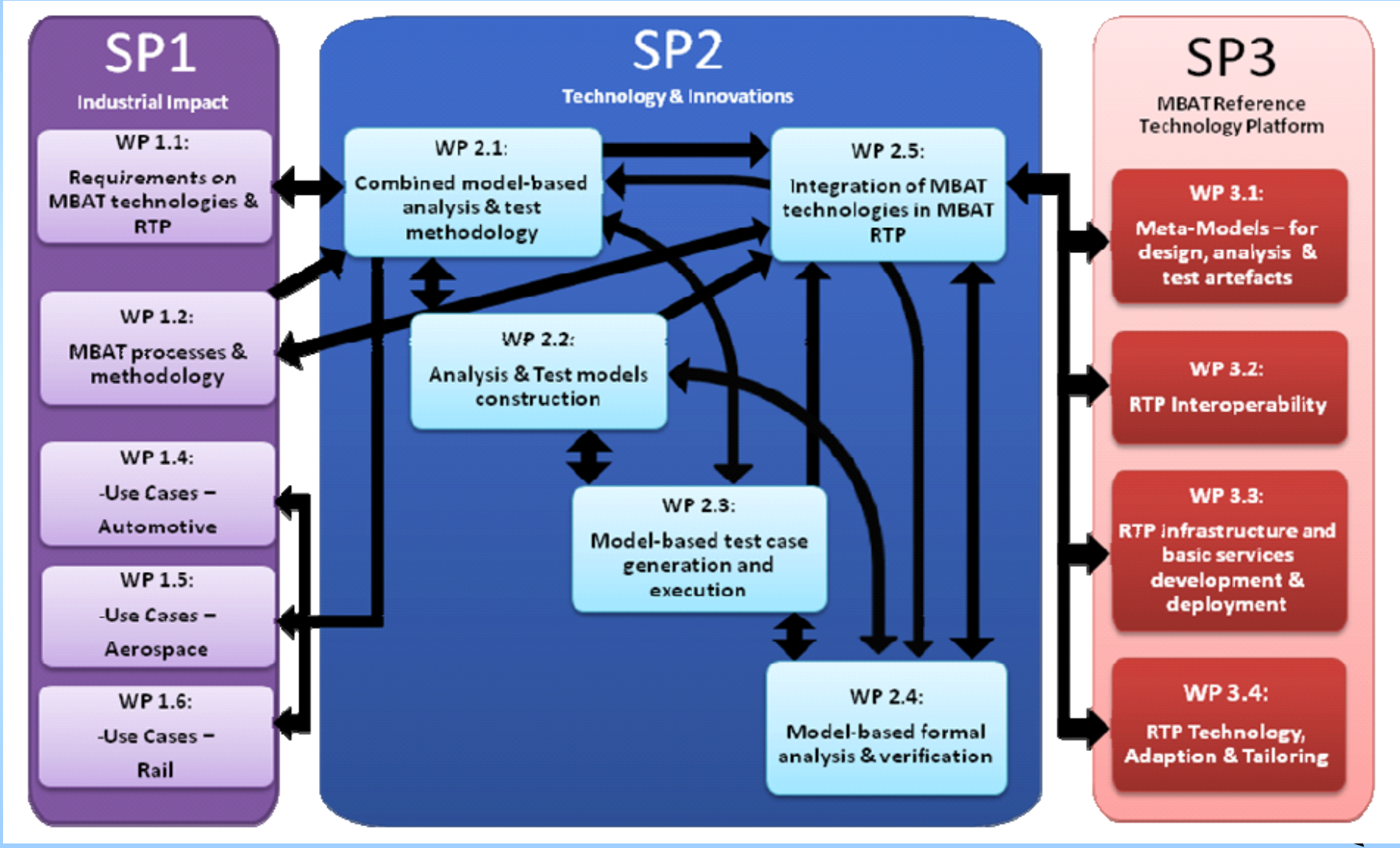
- **MBAT will provide Europe with a new leading-edge *Reference Technology Platform* for effective and cost-reducing Validation and Verification of Embedded Systems**

MBAT Partners

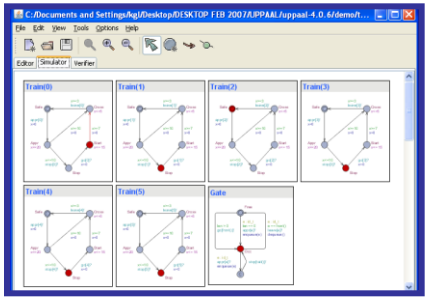


Early Testing at Daimler

MBAT combined methodological approach



QUANTITATIVE Model Checking

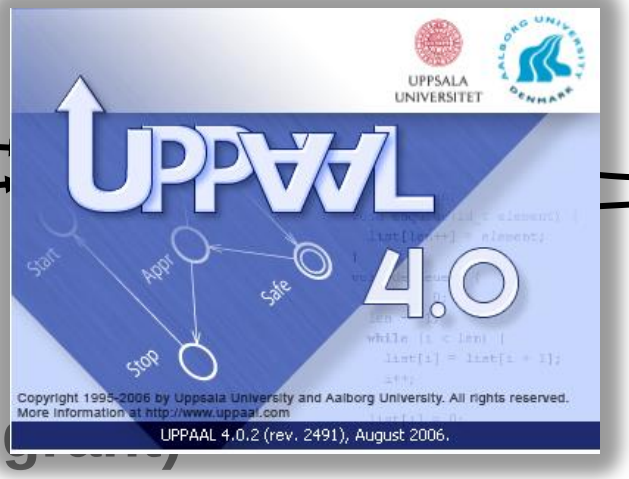


System Description

Time

Cost

Probability



No!
Debugging Information

Yes
Prototypes
Executable Code
Test sequences

Requirement

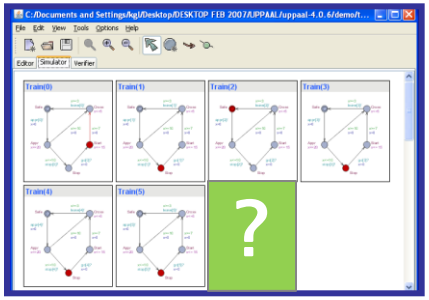
$A\Box(\ req \Rightarrow A\Diamond\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s}\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s,c<5\$}\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s\ ,\ p>0.90}\ grant)$

Synthesis

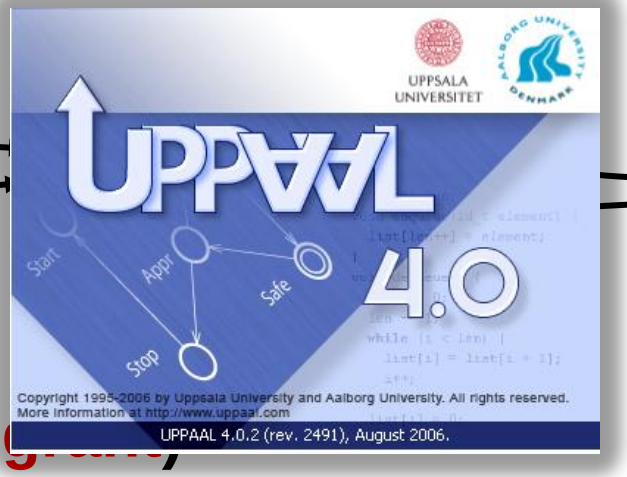


System Description

Time

Cost

Probability



No!

Debugging Information

Yes

Control Strategy

Requirement

$A\Box(\ req \Rightarrow A\Diamond\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s}\ grant)$

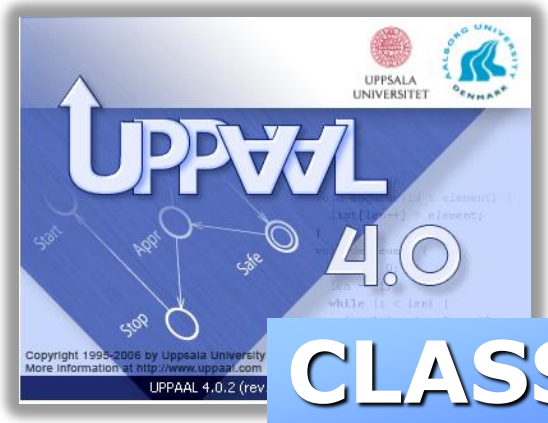
$A\Box(\ req \Rightarrow A\Diamond_{t<30s,c<5\$}\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s\ ,\ p>0.90}\ grant)$

Overview

- Timed Automata & UPPAAL
- Symbolic Verification &
  UPPAAL Engine, Options
- Priced Timed Automata
  and Timed Games
- Stochastic Timed Automata
  Statistical Model Checking

(Lecture + Exercise)^4



CLASSIC

CORA

TIGA

ECDAR

SMC

TRON

www.cs.aau.dk/~kgl/Shanghai2013

Search Facebook Amazon YouTube Weather BBC Ne Options

Model-Based Verification, Optimization, Synthesis and Performance Evaluation of Real-Time Systems

using UPPAAL

www.uppaal.org

Alexandre David & Kim Guldstrand Larsen

CISS, Aalborg University, DENMARK

Timed Automata

AALBORG UNIVERSITY DENMARK

UPPAAL (1995–)

@AALborg

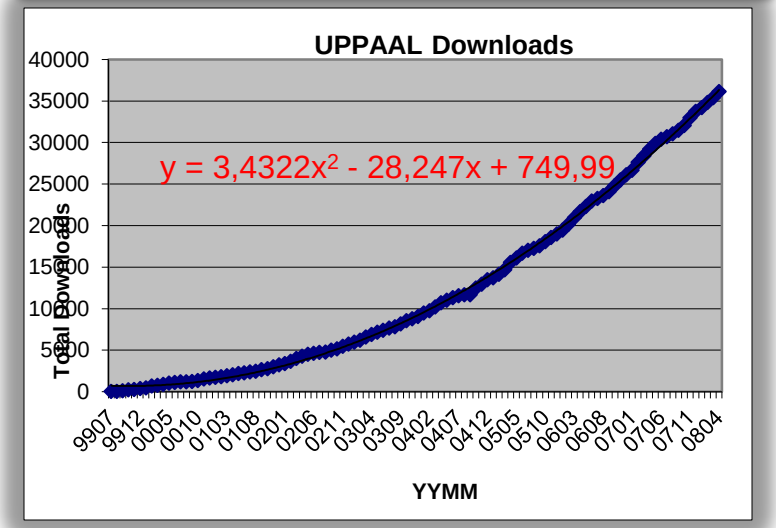
- Kim G Larsen
- Alexandre David
- Gerd Behrman
- Marius Mikucionis
- Jacob I. Rasmussen
- Arne Skou
- Brian Nielsen
- Shuhao Li

@UPPsala

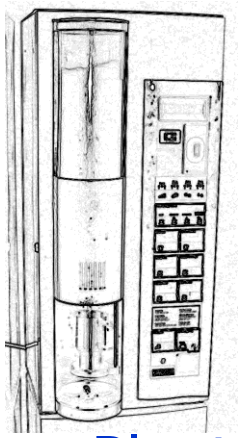
- Wang Yi
- Paul Pettersson
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun

@Elsewhere

Emmanuel Fleury, Didier Lime, Johan Bengtsson, Fredrik Larsson, Kåre J Kristoffersen, Tobias Amnell, Thomas Hune, Oliver Möller, Elena Fersman, Carsten Weise, David Griffioen, Ansgar Fehnker, Jan Tretmans, Frits Vandraager, Theo Ruys, Pedro D'Argenio, J-P Katoen,, Judi Romijn, Ed Brinksma, Martijn Hendriks, Klaus Havelund, Franck Cassez, Magnus Lindahl, Francois Laroussinie, Patricia Bouyer, Augusto Burgueno, H. Bowmann, D. Latella, M. Massink, G. Faconti, Kristina Lundqvist, Lars Asplund, Justin Pearson.....

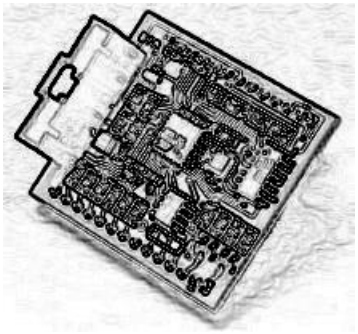


Real Time Systems



sensors

actuators



Plant

Continuous

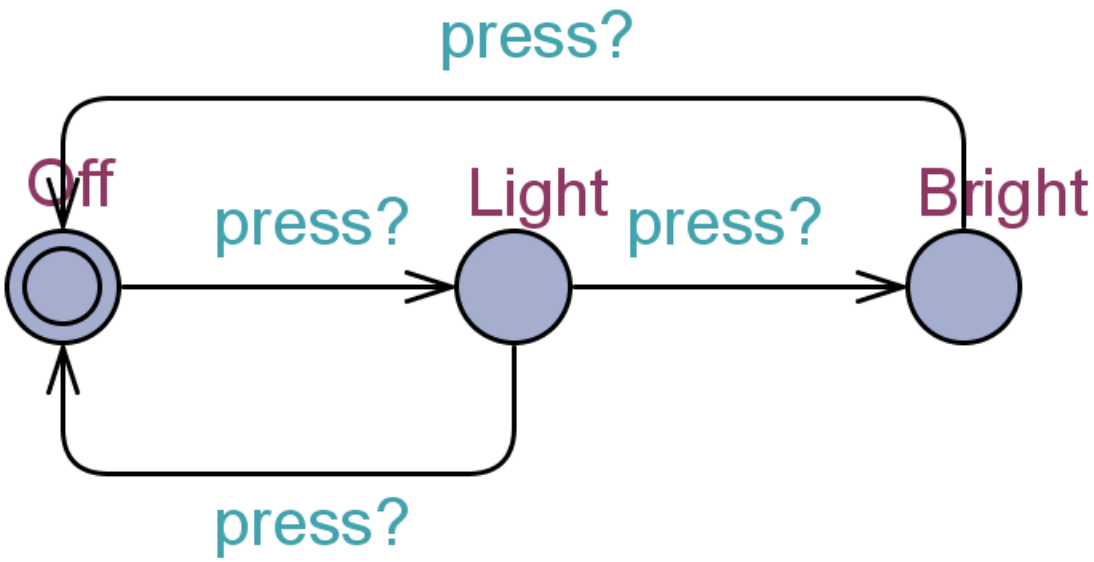
Controller Program

Discrete

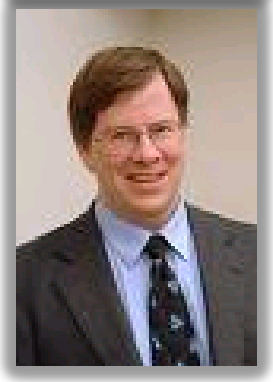
Eg.: Realtime Protocols
Pump Control
Air Bags
Robots
Cruise Control
ABS
CD Players
Production Lines

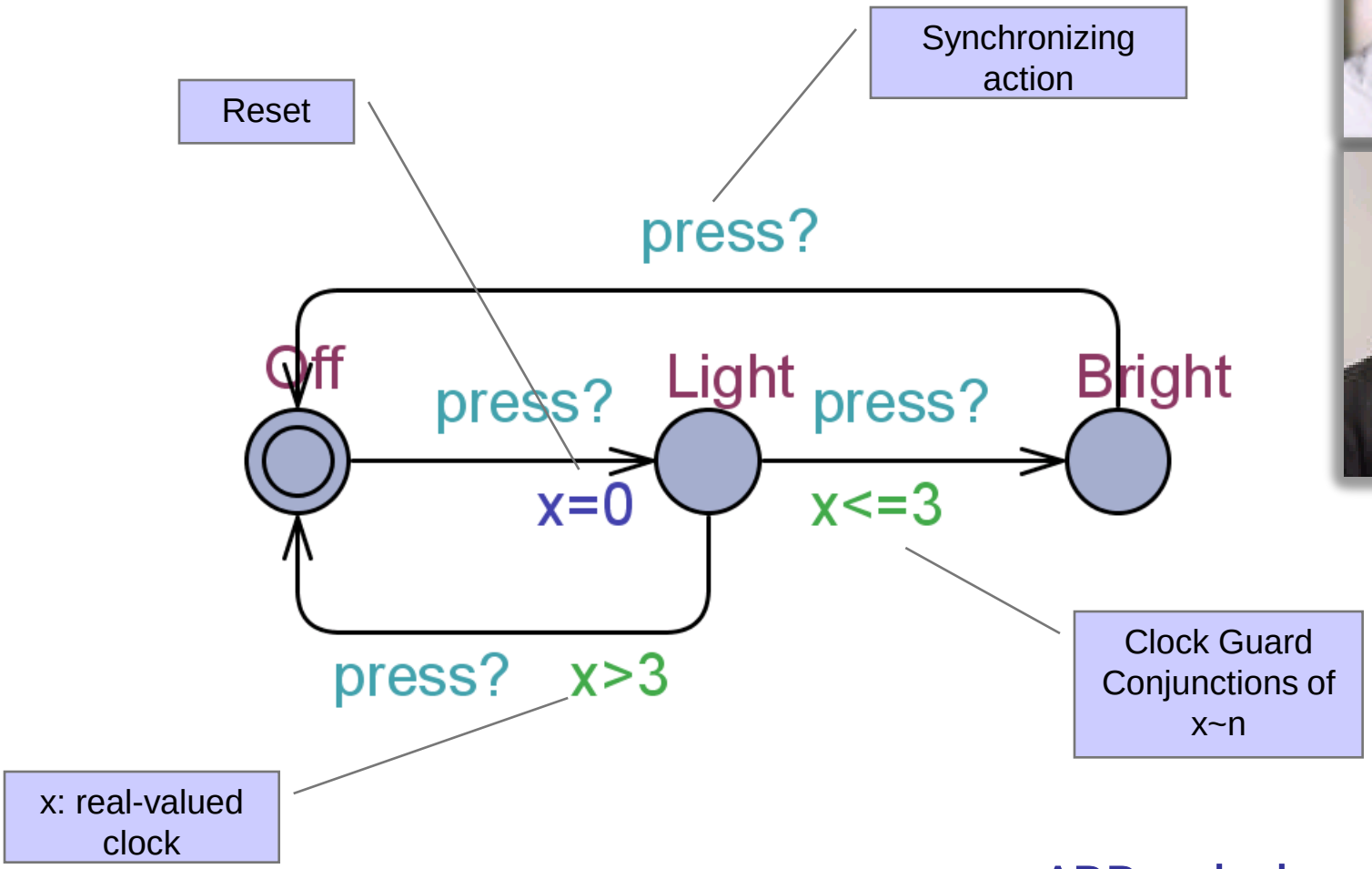
Real Time System
A system where correctness not only depends on the logical order of events but also on their timing!!

A Dumb Light Controller



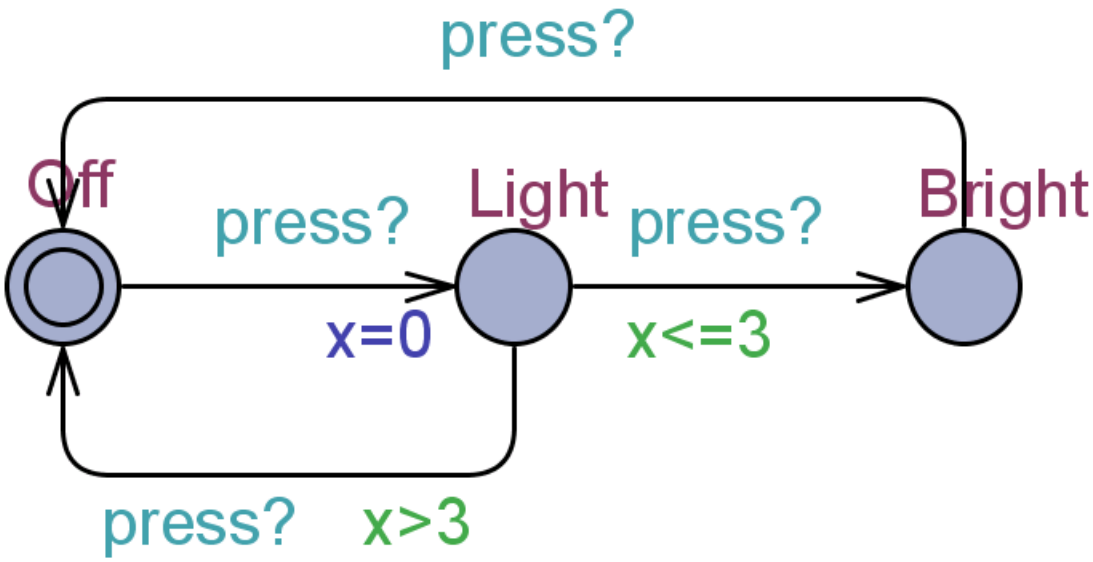
Timed Automata [Alur & Dill'89]





ADD a clock x

A Timed Automata (Semantics)

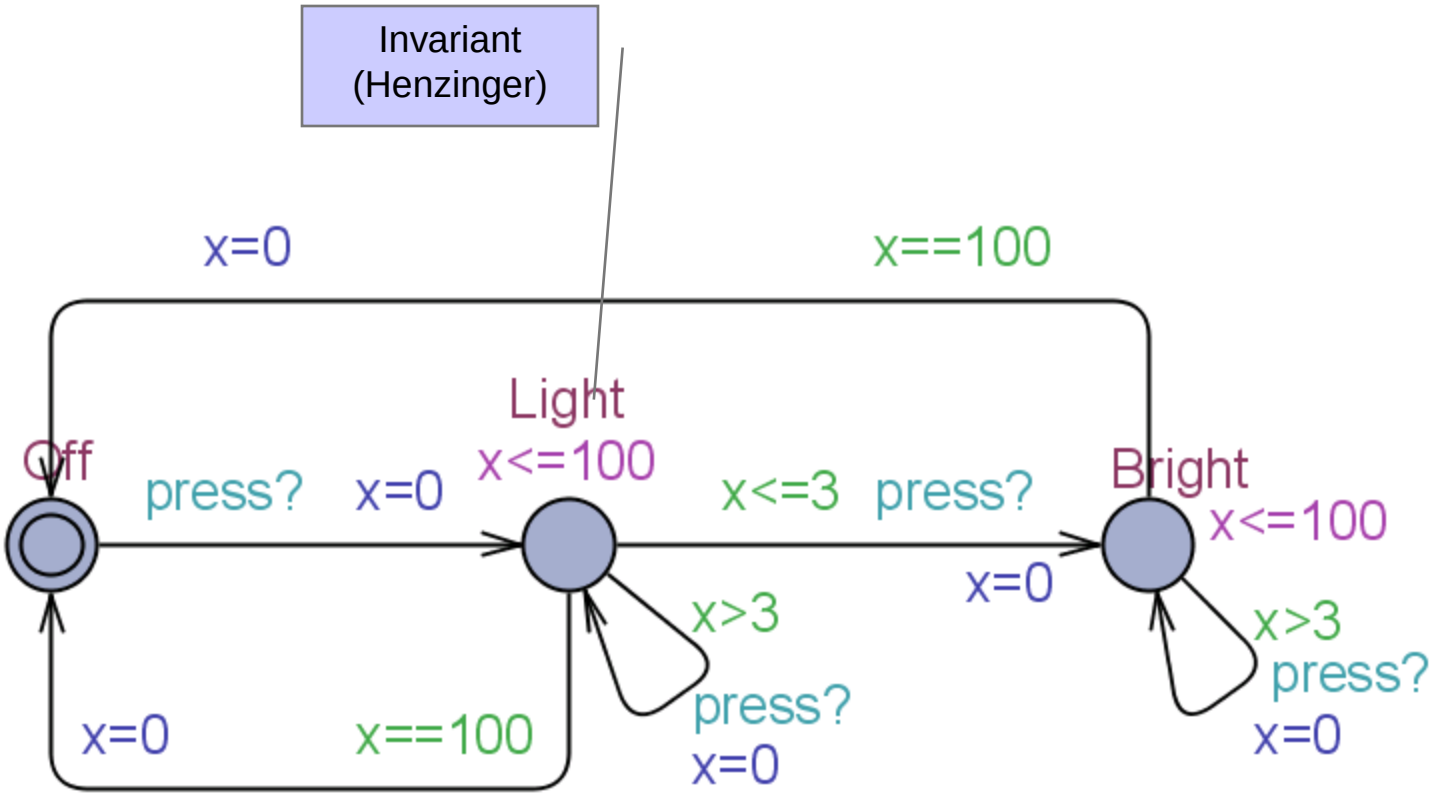


States:
(location , x=v) where v∈**R**

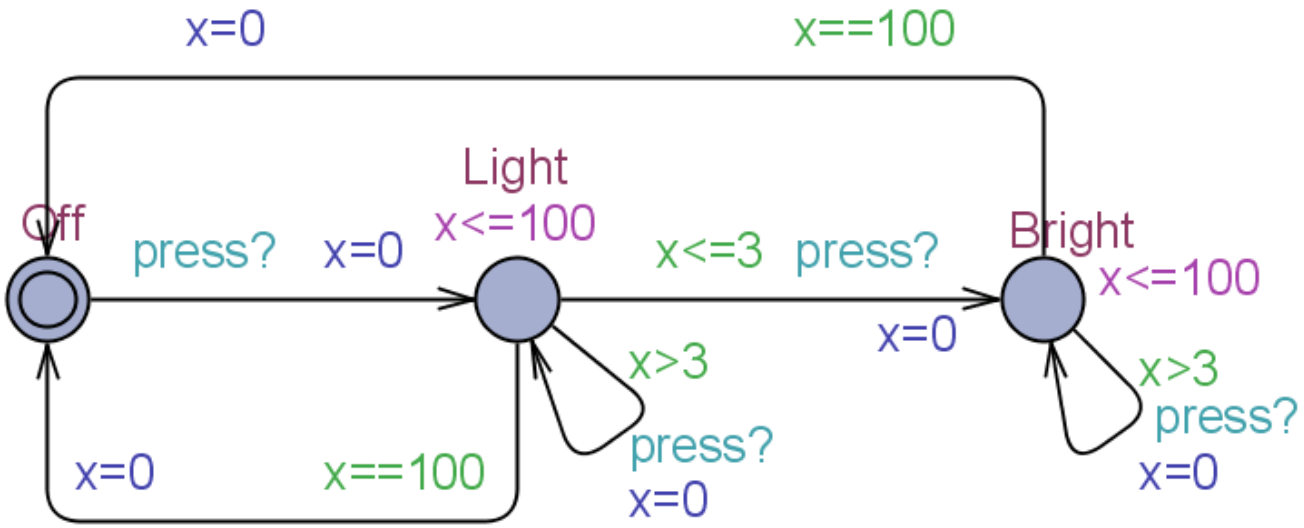
Transitions:

	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 2.51	→ (Light , x=2.51)
press?	→ (Bright , x=2.51)

Intelligent Light Controller



Intelligent Light Controller



Transitions:

	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 4.51	→ (Light , x=4.51)
press?	→ (Light , x=0)
delay 100	→ (Light , x=100)
τ	→ (Off , x=0)

Note:

(Light , x=0) delay 103 →

X

Invariants ensures progress

Timed Automata (formally)

Constraints

Definition

Let X be a set of clock variables. The set $\mathcal{B}(X)$ of *clock constraints* ϕ is given by the grammar:

$$\phi \ ::= \ x \leq c \mid c \leq x \mid x < c \mid c < x \mid \phi_1 \wedge \phi_2$$

where $c \in \mathbb{N}$ (or $\mathbb{Q}$).

Timed Automata (formally)

Clock Valuations and Notation

Definition

The set of *clock valuations*, $\mathbb{R}^C$ is the set of functions $C \longrightarrow \mathbb{R}_{\geq 0}$ ranged over by $u, v, w, \ldots$.

Notation

Let $u \in \mathbb{R}^C$, $r \subseteq C$, $d \in \mathbb{R}_{\geq 0}$, and $g \in \mathcal{B}(X)$ then:

- $u + d \in \mathbb{R}^C$ is defined by $(u + d)(x) = u(x) + d$ for any clock x
- $u[r] \in \mathbb{R}^C$ is defined by $u[r](x) = 0$ when $x \in r$ and $u[r](x) = u(x)$ for $x \notin r$.
- $u \models g$ denotes that g is satisfied by u.

Timed Automata (formally)

Timed Automata

Definition

A timed automaton A over clocks C and actions Act is a tuple (L, l_0, E, I), where:

- L is a finite set of locations
- $l_0 \in L$ is the initial location
- $E \subseteq L \times \mathcal{B}(X) \times Act \times \mathcal{P}(C) \times L$ is the set of edges
- $I : L \longrightarrow \mathcal{B}(X)$ assigns to each location an invariant

Timed Automata (formally)

Semantics

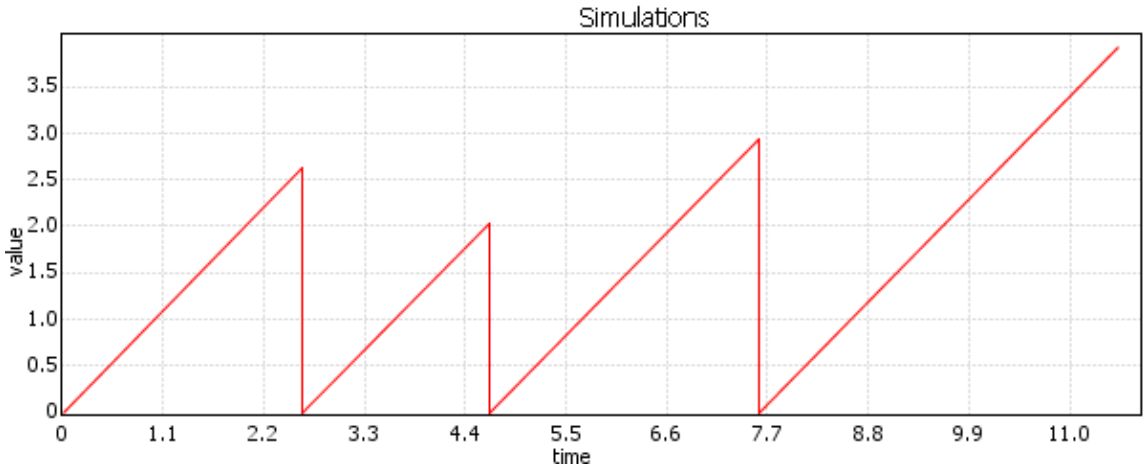
Definition

The semantics of a timed automaton A is a labelled transition system with state space $L \times \mathbb{R}^C$ with initial state $(l_0, u_0)^*$ and with the following transitions:

- $(l,u) \xrightarrow{\epsilon(d)} (l,u+d)$ iff $u \in I(l)$ and $u+d \in I(l)$,
- $(l,u) \xrightarrow{a} (l',u')$ iff there exists $(l,g,a,r,l') \in E$ such that
  - $u \models g$,
  - $u' = u[r]$, and
  - $u' \in I(l')$

$^*u_0(x) = 0$ for all $x \in C$

Timed Automata: Example



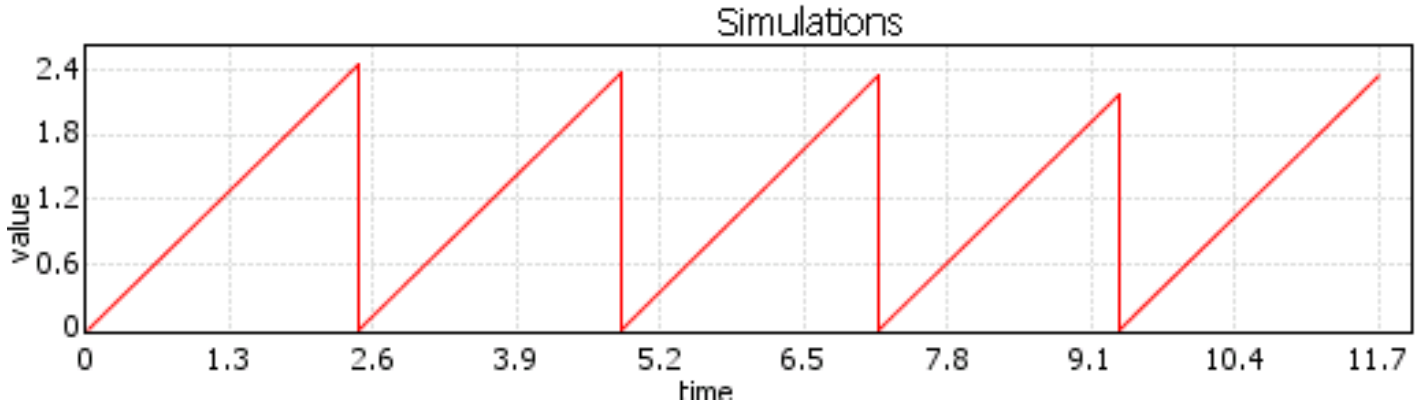
guard

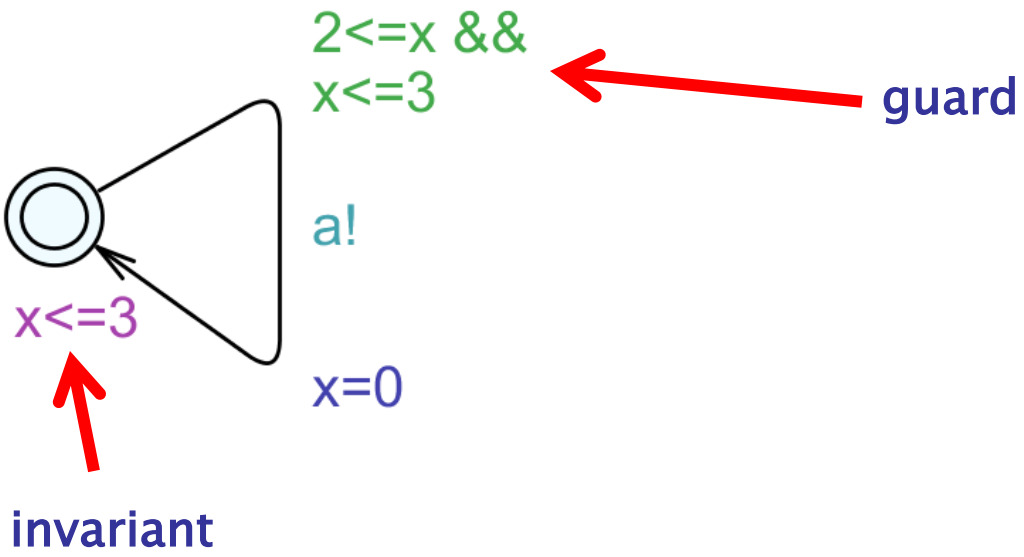
2<=x &&
x<=3

a! ← synchronization

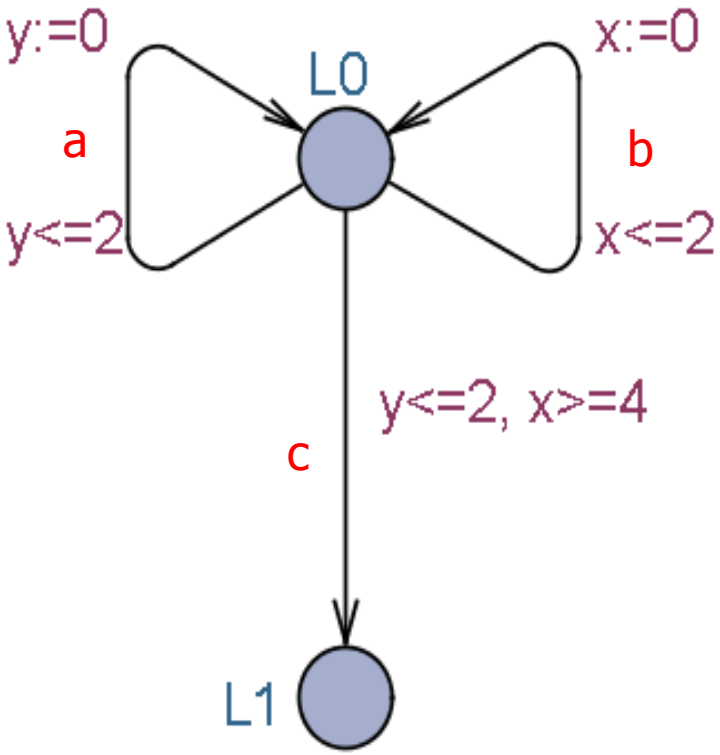
x=0 ← reset

Timed Automata: Example



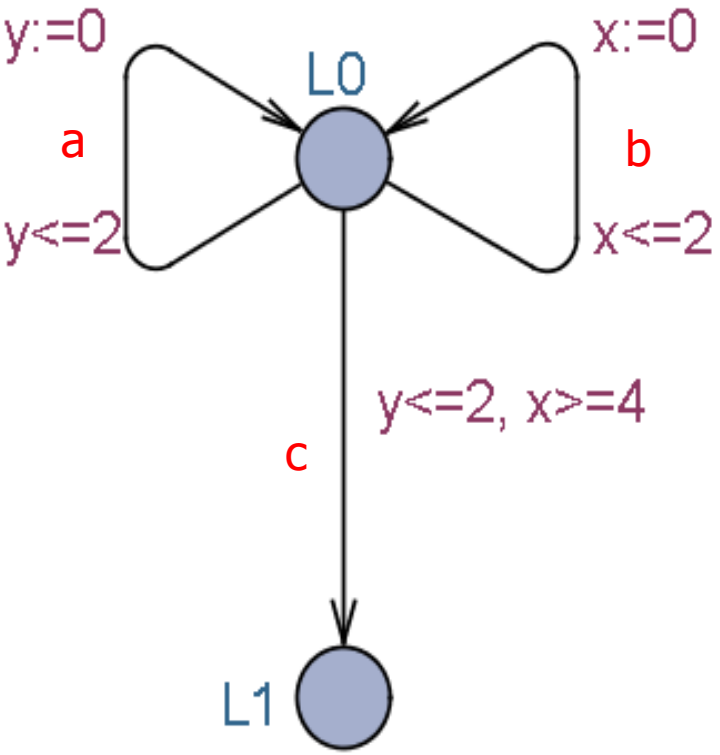


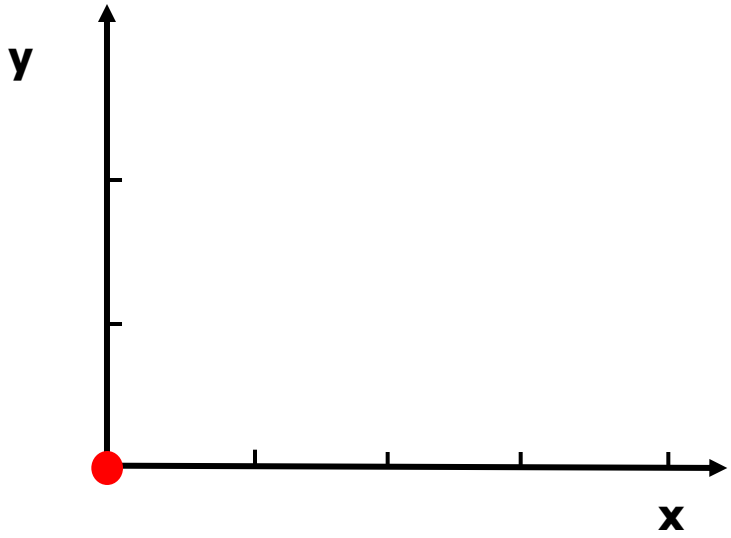
Example



Is L1 reachable ?

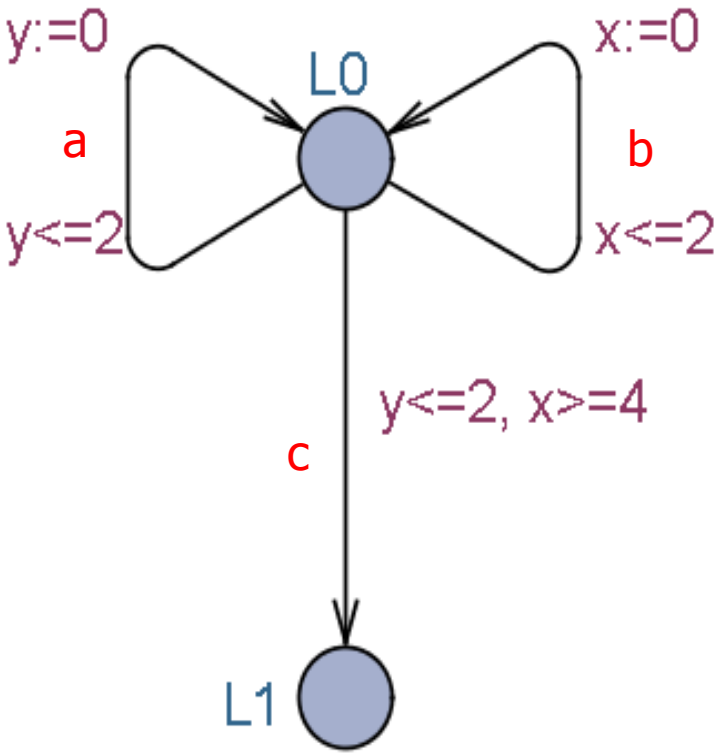
Example

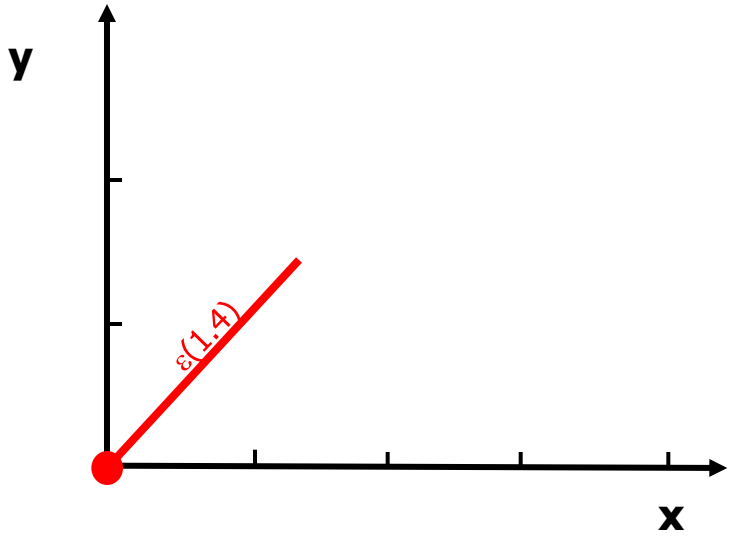




$(\ell_0, x = 0, y = 0)$

Example

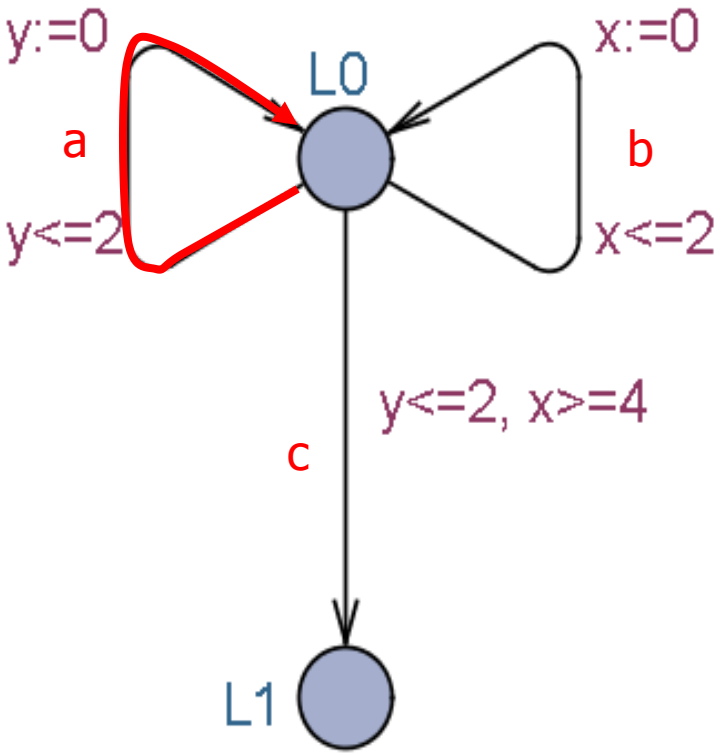


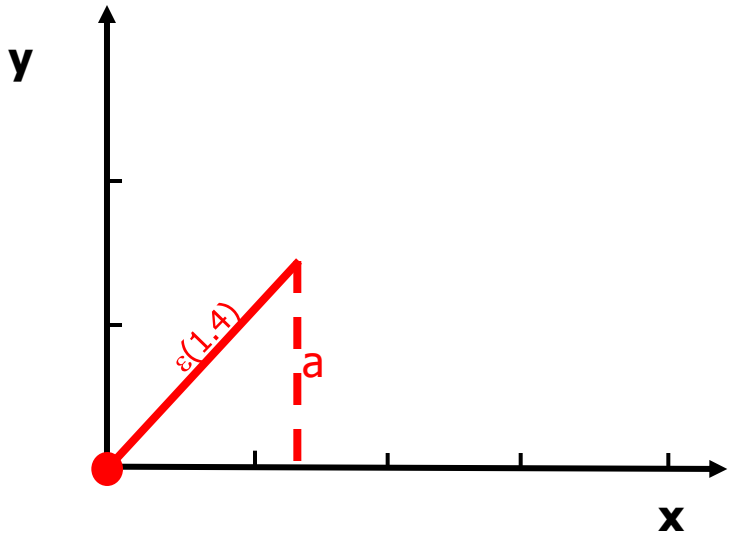


$$(\ell_0, x = 0, y = 0)$$

$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$

Example



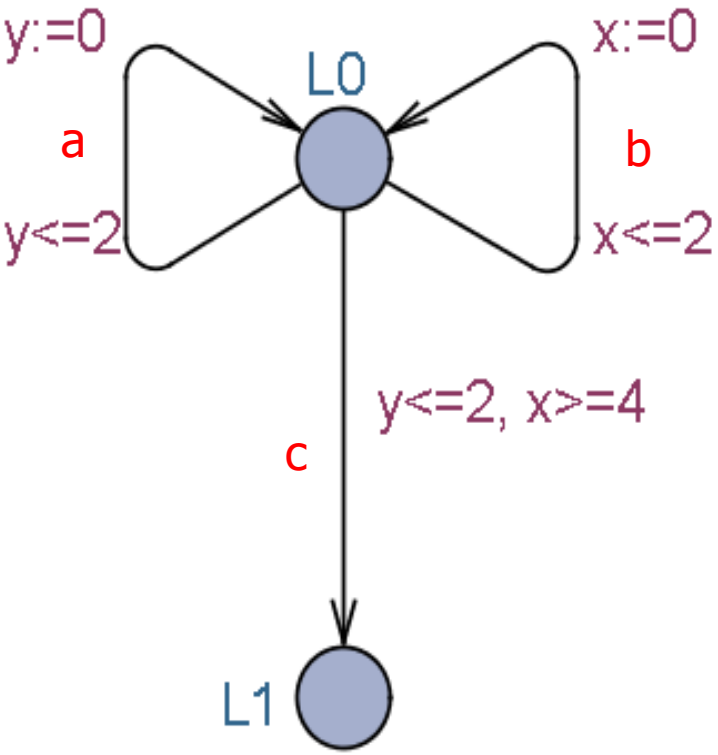


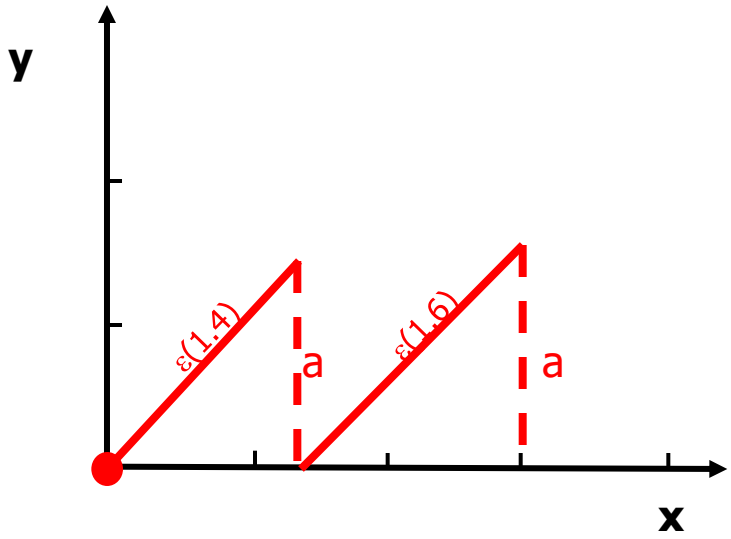
$$(\ell_0, x = 0, y = 0)$$

$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$

$$\xrightarrow{a} (\ell_0, x = 1.4, y = 0)$$

Example





$$(\ell_0, x = 0, y = 0)$$

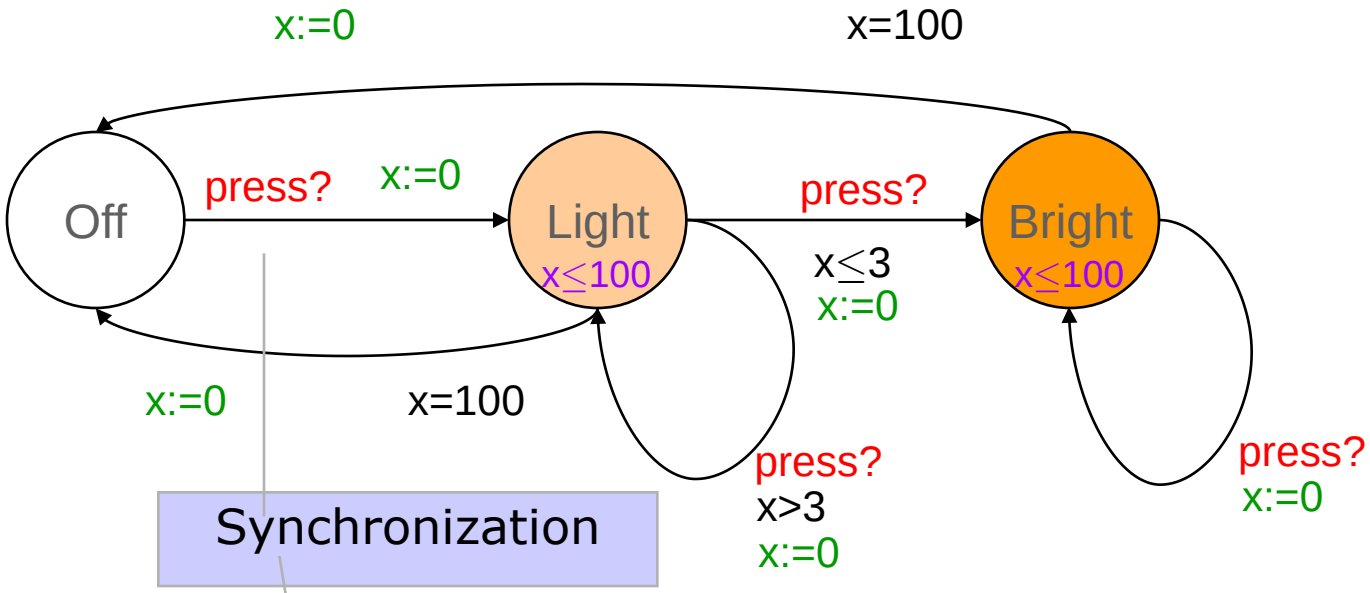
$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$

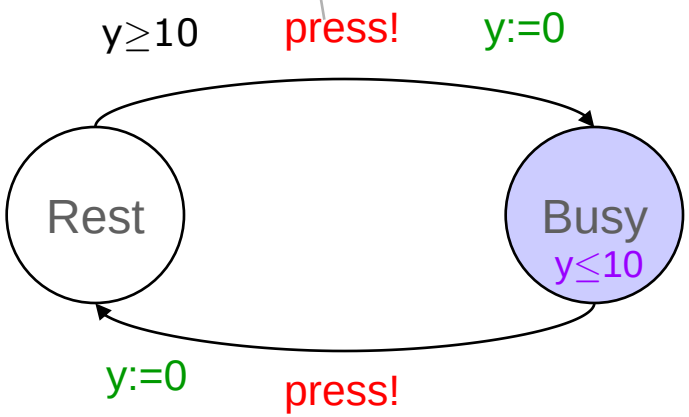
$$\xrightarrow{a} (\ell_0, x = 1.4, y = 0)$$

$$\xrightarrow{1.6} (\ell_0, x = 3.0, y = 1.6)$$

$$\xrightarrow{a} (\ell_0, x = 3.0, y = 0)$$

Networks Light Controller & User





Transition

(Off, Rest, x=0, y=0)

delay 20 → (Off, Rest, x=20, y=20)

press?! → (Light, Busy, x=0, y=0)

delay 2 → (Light, Busy, x=2, y=2)

press?! → (Bright, Rest, x=0, y=0)

Network Semantics

$$T_1 \|_X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \|_X s_0^2)$$ where

$$\frac{s_1 \xrightarrow{\mu}_1 s_1'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1' \|_X s_2} \qquad \frac{s_2 \xrightarrow{\mu}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1 \|_X s_2'}$$

$$\frac{s_1 \xrightarrow{a!}_1 s_1' \qquad s_2 \xrightarrow{a?}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\tau} s_1' \|_X s_2'}$$

$$\frac{s_1 \xrightarrow{e(d)}_1 s_1' \qquad s_2 \xrightarrow{e(d)}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{e(d)} s_1' \|_X s_2'}$$

Network Semantics

(URGENT synchronization)

+ Urgent synchronization

$$T_1 \|_X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \|_X s_0^2)$$ where

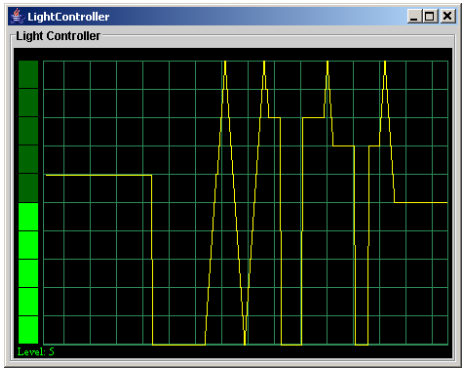
$$\frac{s_1 \xrightarrow{\mu}_1 s_1'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1' \|_X s_2} \qquad \frac{s_2 \xrightarrow{\mu}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1 \|_X s_2'}$$

$$\frac{s_1 \xrightarrow{a!}_1 s_1' \qquad s_2 \xrightarrow{a?}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\tau} s_1' \|_X s_2'}$$

$\forall d' < d, \forall u \in UAct:$

$\neg (s_1 \xrightarrow{e(d')} \xrightarrow{u?} \wedge s_2 \xrightarrow{e(d')} \xrightarrow{u!})$

$$\frac{s_1 \xrightarrow{e(d)}_1 s_1' \qquad s_2 \xrightarrow{e(d)}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{e(d)} s_1' \|_X s_2'}$$



Light Control Interface

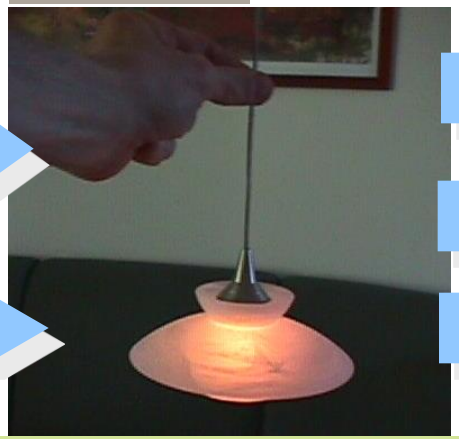
Light Control Interface

press? 0.2 release? … press? 0.7 release? … press? 1.0 2.4 release? …

↓ Ø ↓ touch! ↓ starthold! ↓ endhold!

User

Interface



press?

release?

touch!

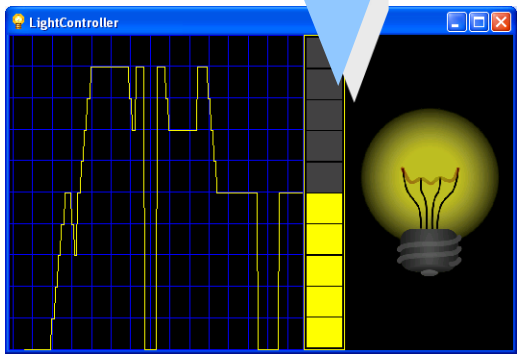
starthold!

endhold!

Control Program

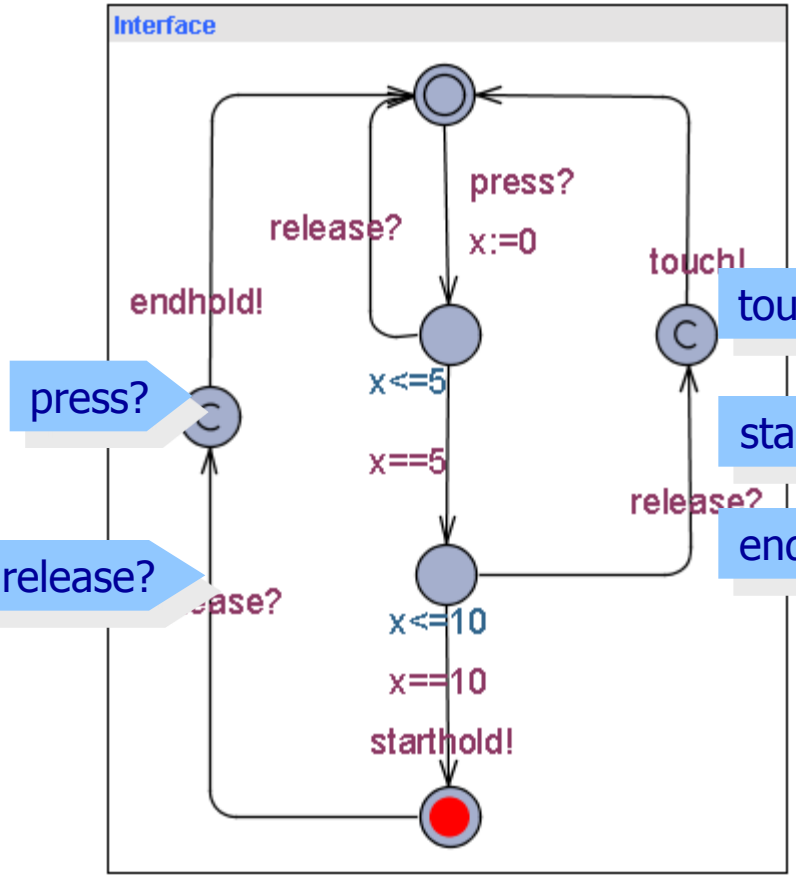
L++/L--/L:=0

press? d release? → touch! 0.5≤d≤ 1
press? 1 → starthold!
press? d release? → endhold! d >1



Light Control Interface

User

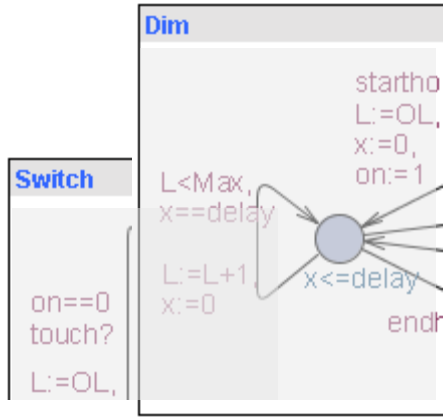


touch!

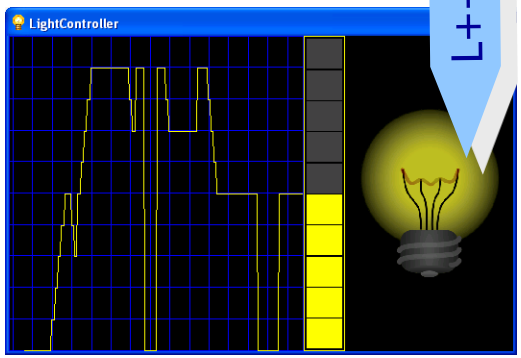
starthold!

endhold!

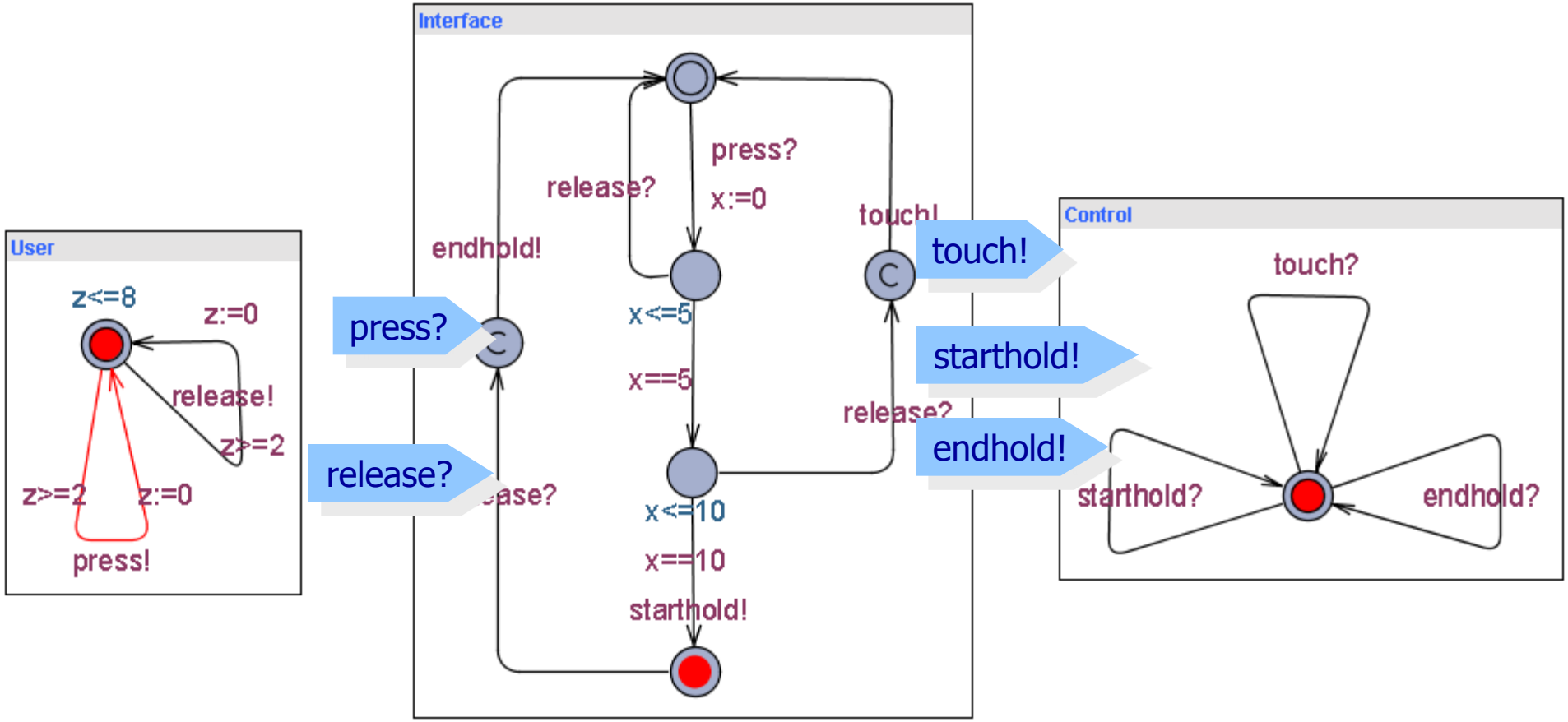
Control Program



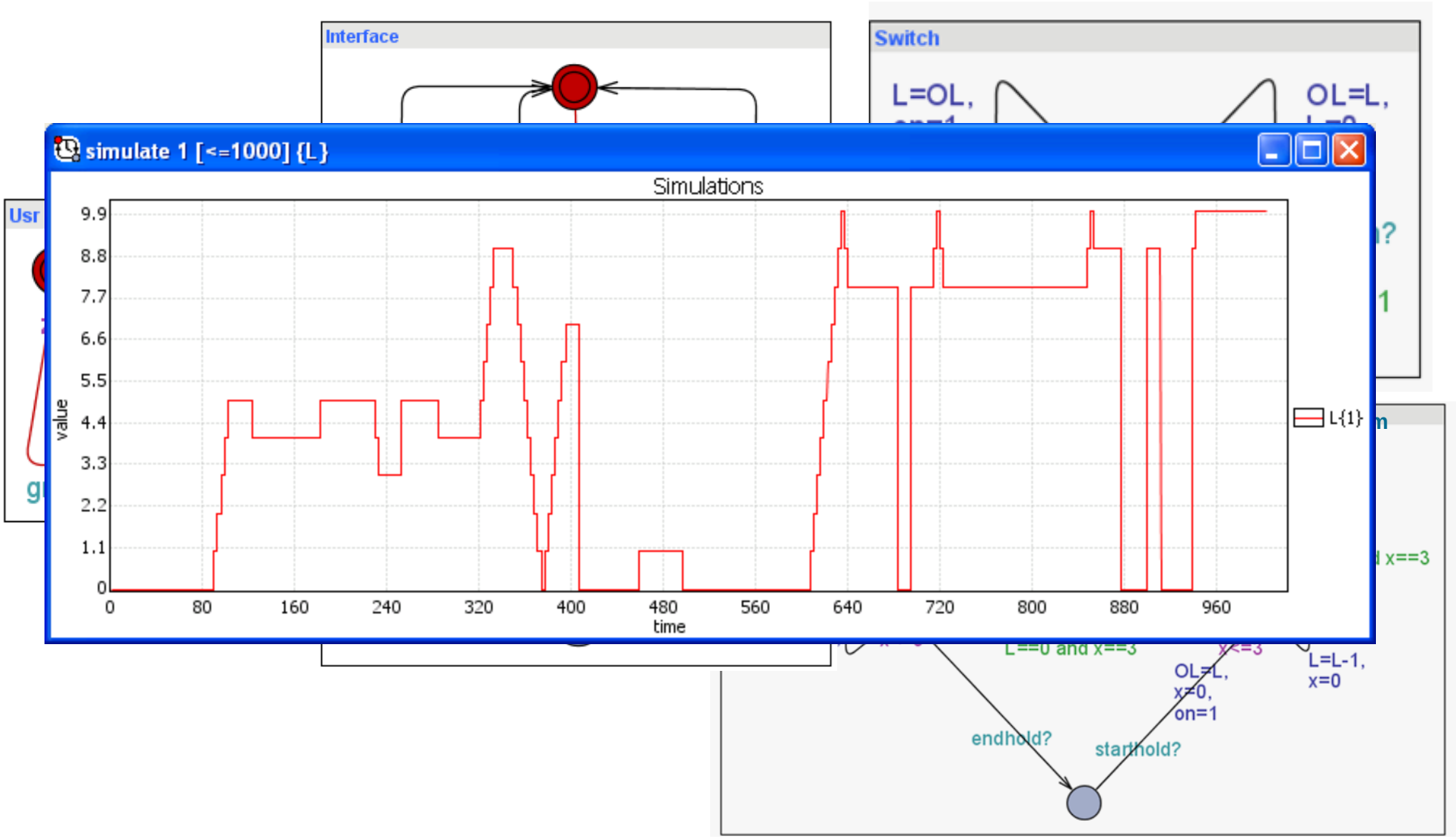
L++/L--/L:=0



Light Control Network



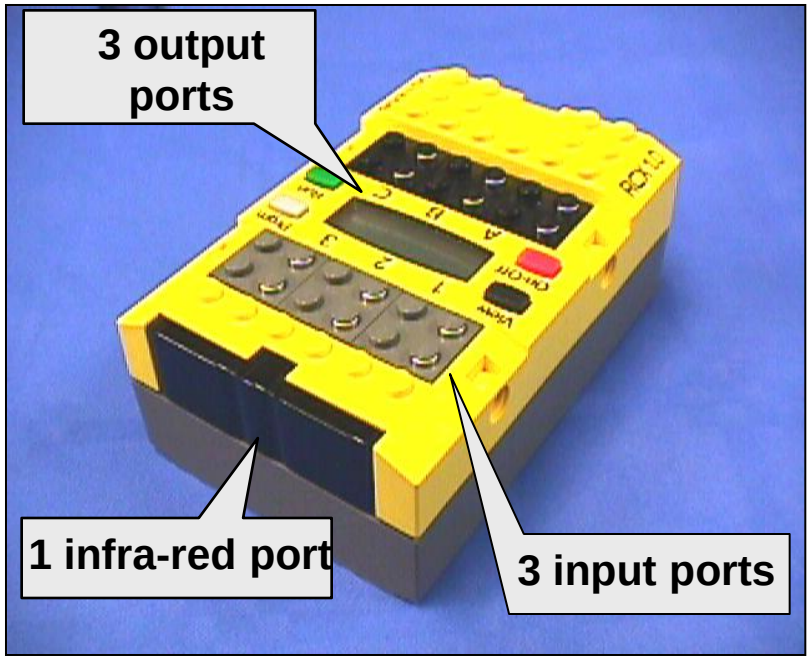
Full Light Controller



Brick Sorting

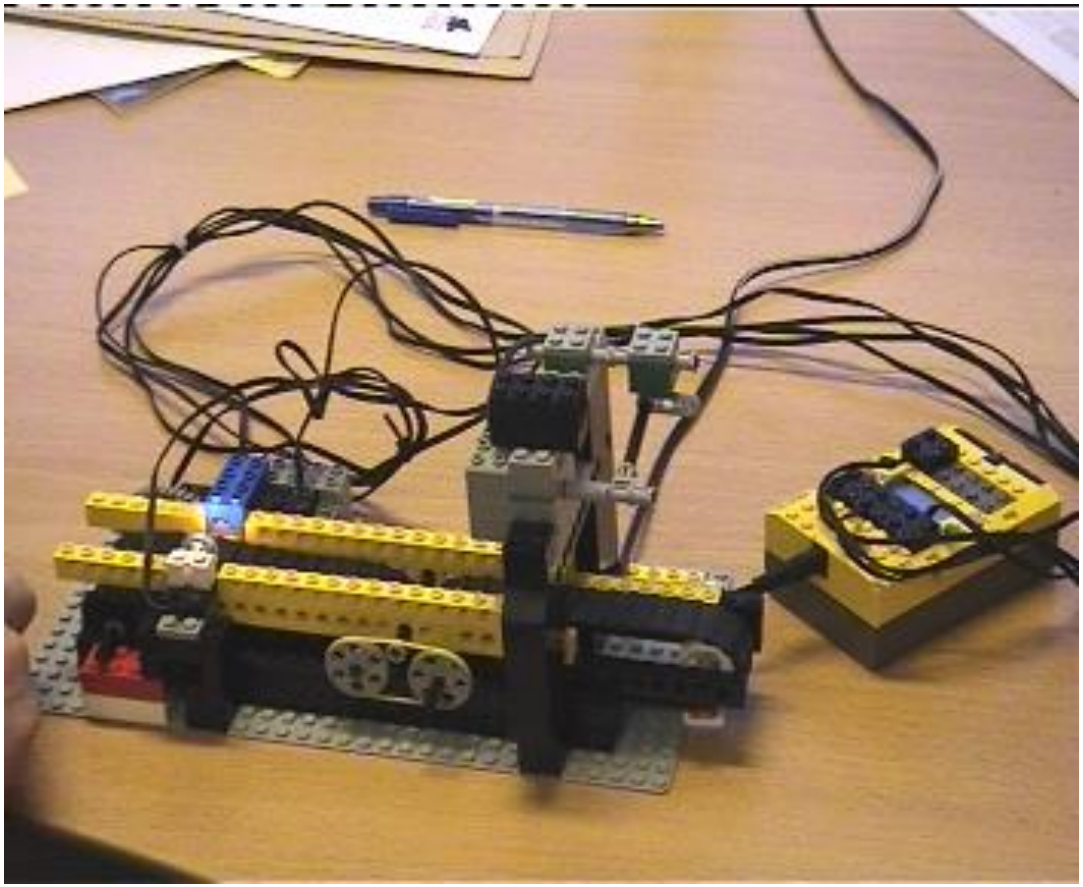
LEGO Mindstorms/RCX

- Sensors: temperature, light, rotation, pressure.
- Actuators: motors, lamps,
- Virtual machine:
  - 10 tasks, 4 timers, 16 integers.
- Several Programming Languages:
  - NotQuiteC, Mindstorm, Robotics, legOS, etc.



A Real Real Timed System

The Plant
Conveyor Belt
&
Bricks

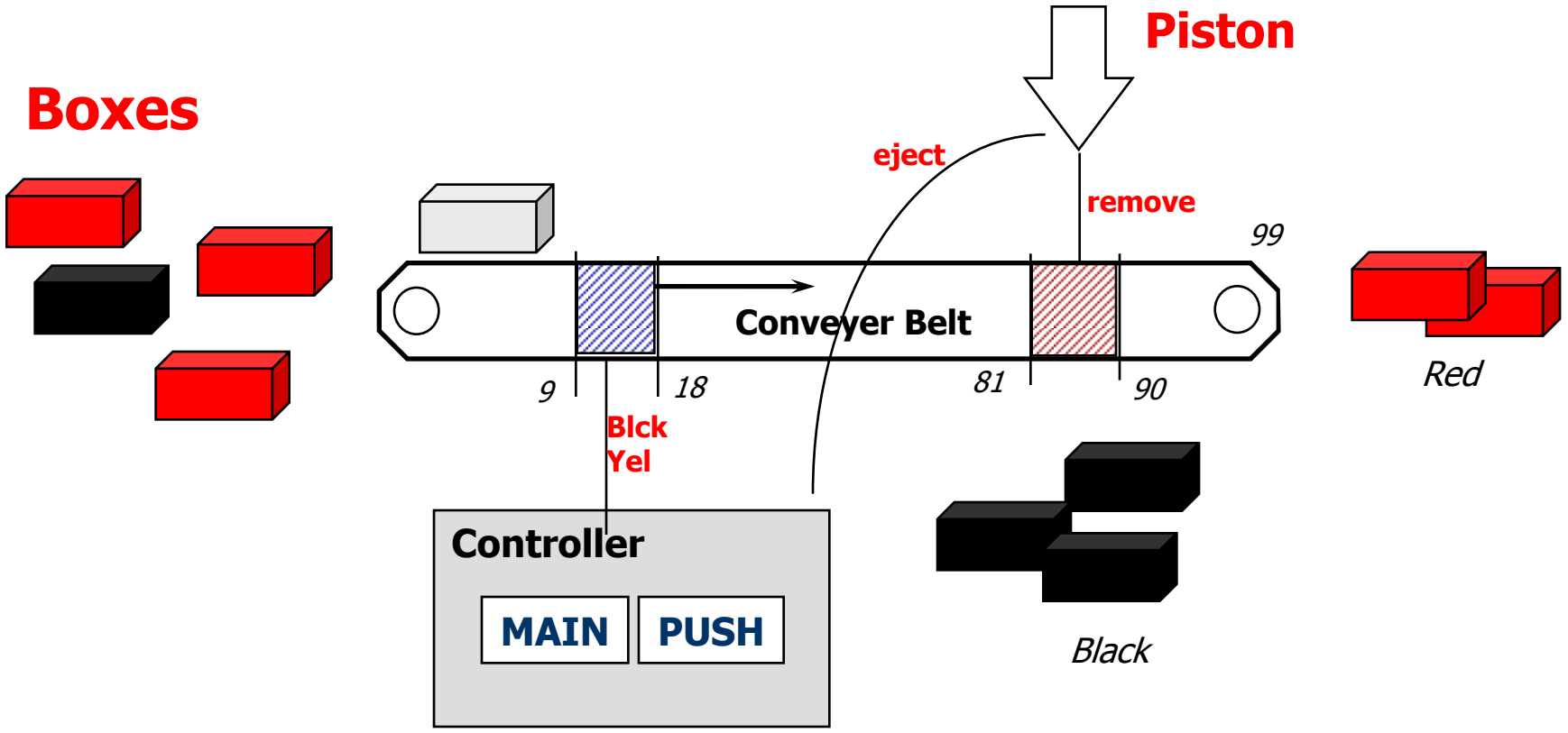


Controller Program
LEGO MINDSTORM

First UPPAAL model

Sorting of Lego Boxes

Ken Tindell



Exercise: Design **Controller** so that **black** boxes are being pushed out

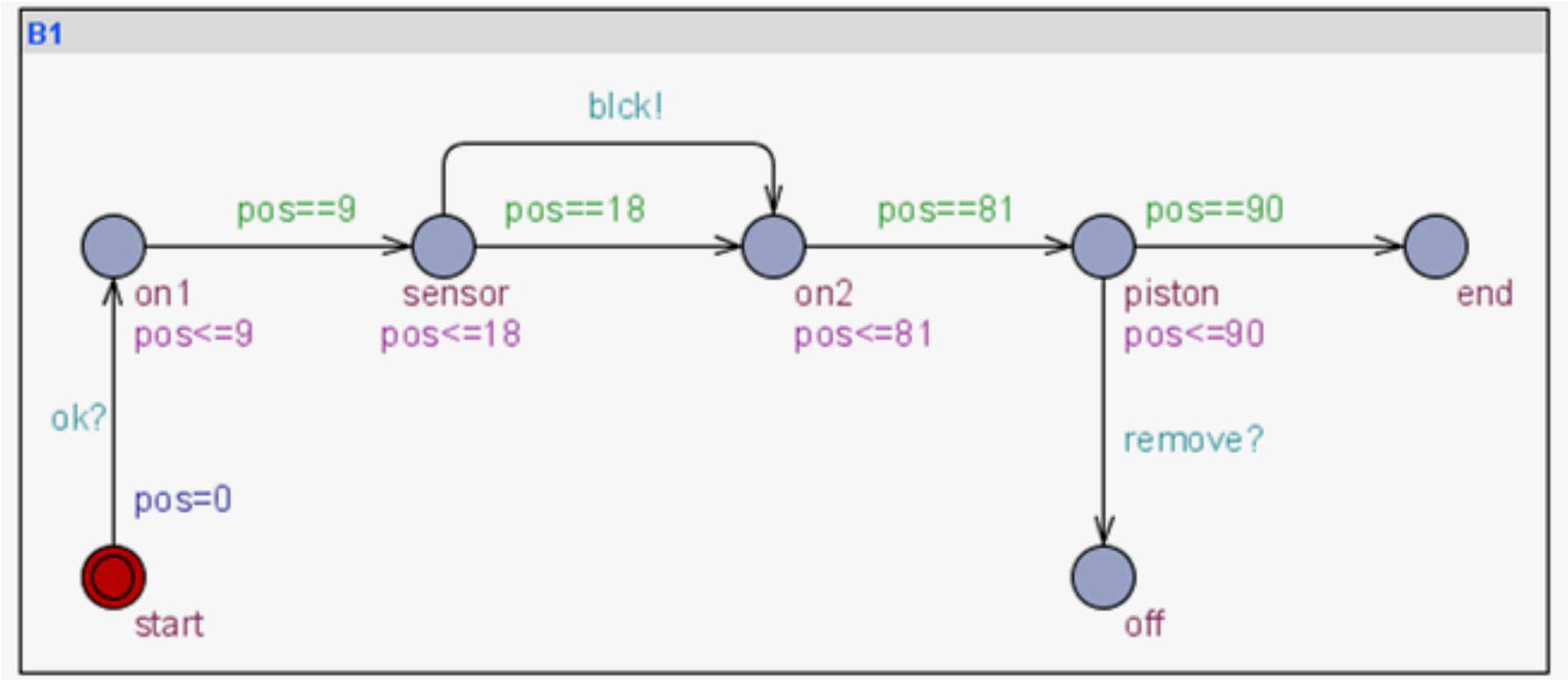
NQC programs

```
int active;
int DELAY;
int LIGHT_LEVEL;
```

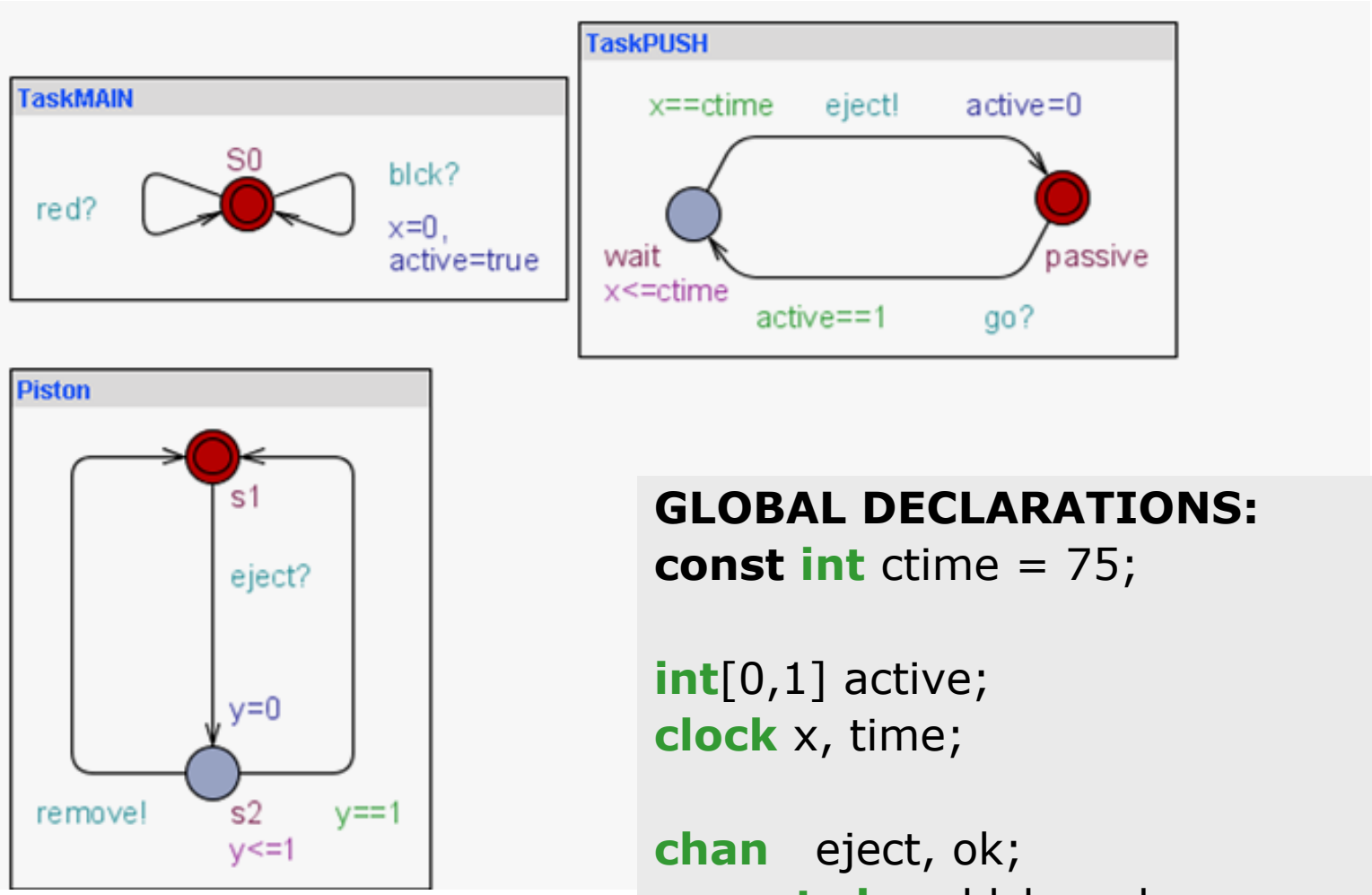
```
task MAIN{
 DELAY=75;
 LIGHT_LEVEL=35;
 active=0;
 Sensor(IN_1, IN_LIGHT);
 Fwd(OUT_A,1);
 Display(1);

 start PUSH;

 while(true){

wait(IN_1<=LIGHT_LEVEL);
   ClearTimer(1);
   active=1;
   PlaySound(1);

wait(IN_1>LIGHT_LEVEL);
  }
}
```

```
task PUSH{
  while(true){
    wait(Timer(1)>DELAY && active==1);
    active=0;
    Rev(OUT_C,1);
    Sleep(8);
    Fwd(OUT_C,1);
    Sleep(12);
    Off(OUT_C);
  }
}
```

A Black Brick



Control Tasks & Piston



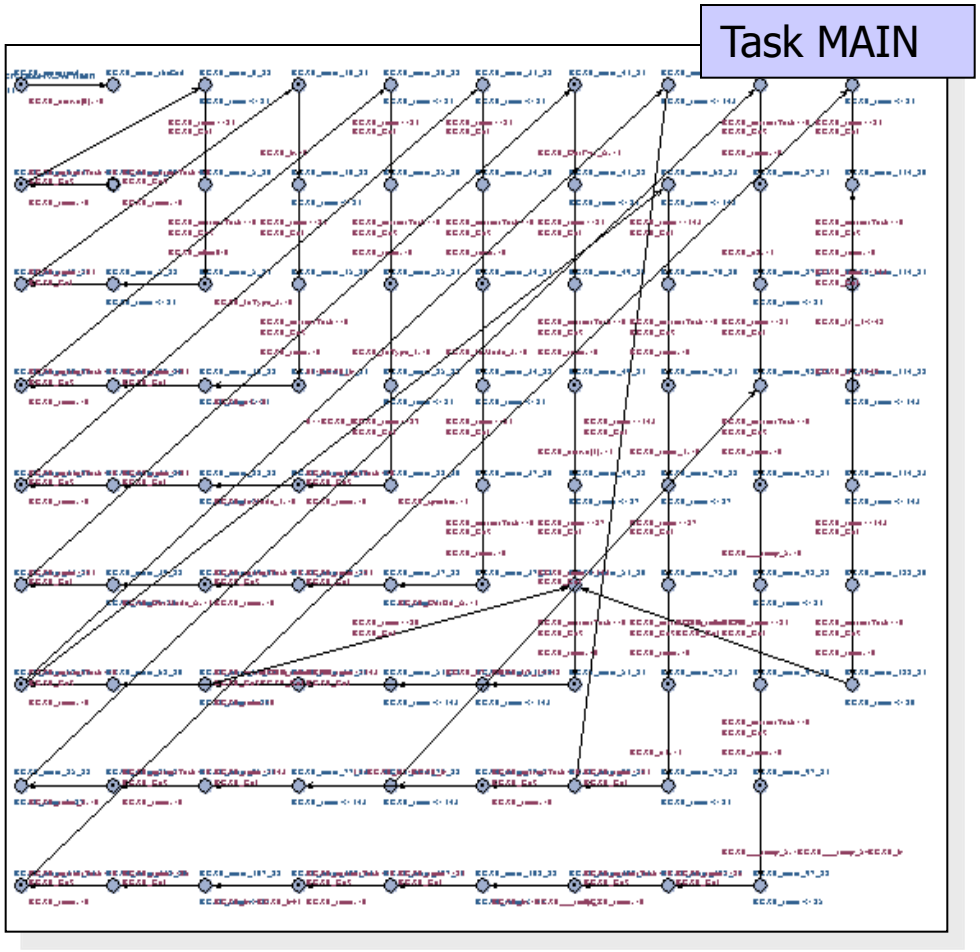
GLOBAL DECLARATIONS:

```
const int ctime = 75;

int[0,1] active;
clock x, time;

chan   eject, ok;
urgent chan blck, red, remove, go;
```

From RCX to UPPAAL – and back

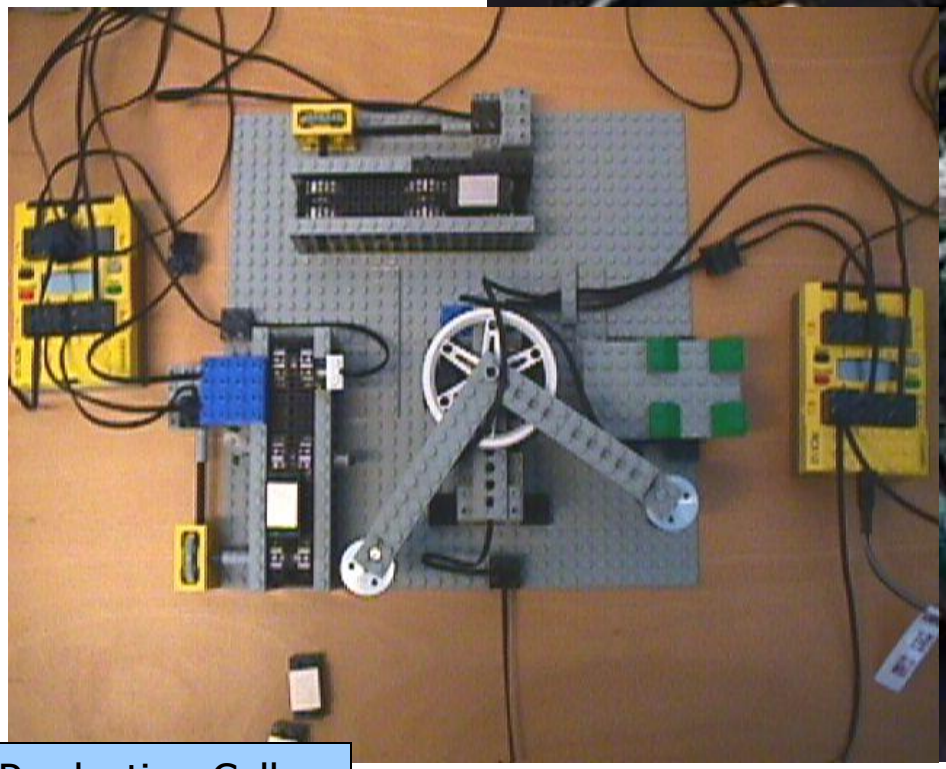
- Model includes Round-Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000 in Stockholm.

- From UPPAAL to RCX: Martijn Hendriks.

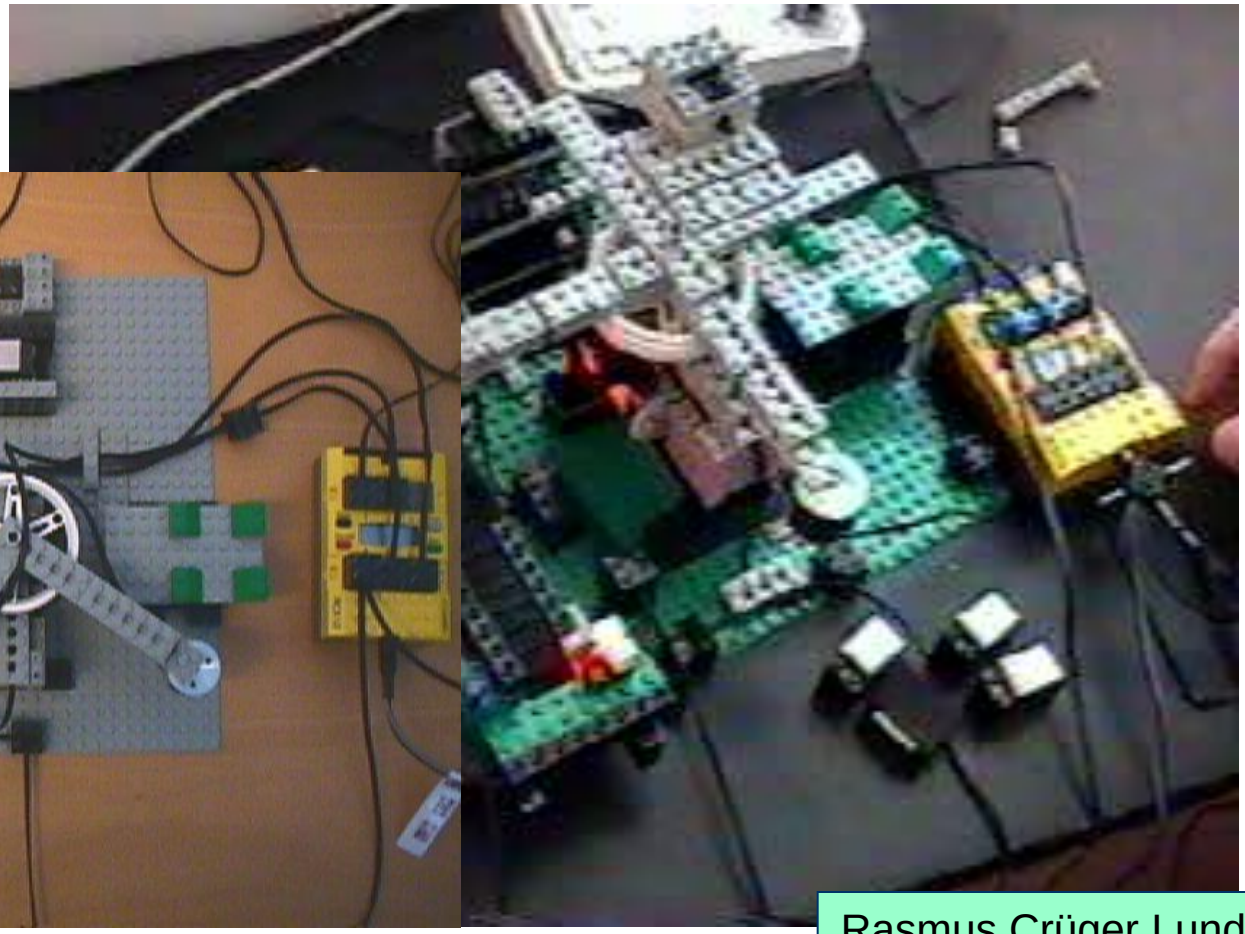


The Production Cell in LEGO

Course at DTU, Copenhagen



Production Cell

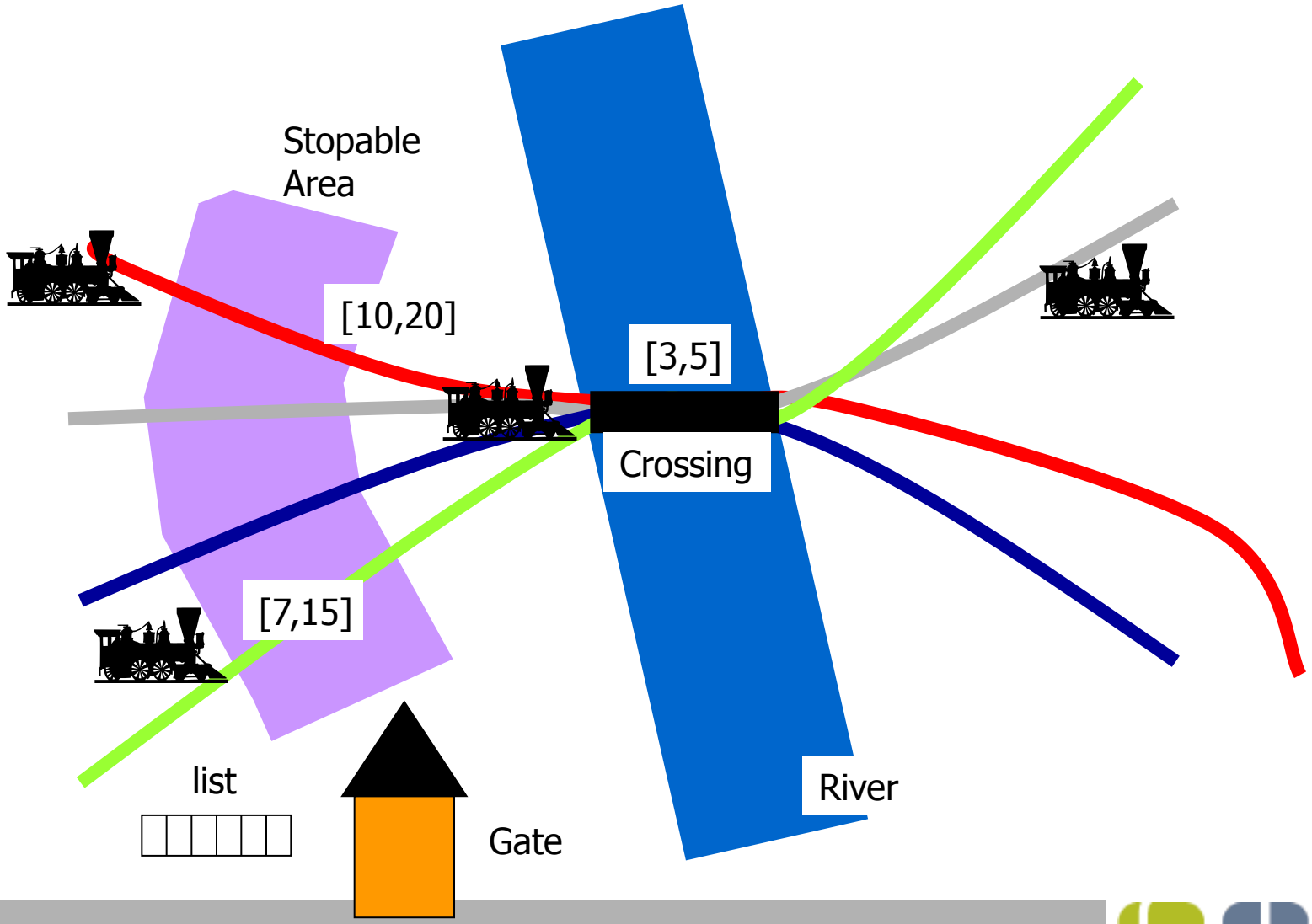


Rasmus Crüger Lund
Simon Tune Riemanni

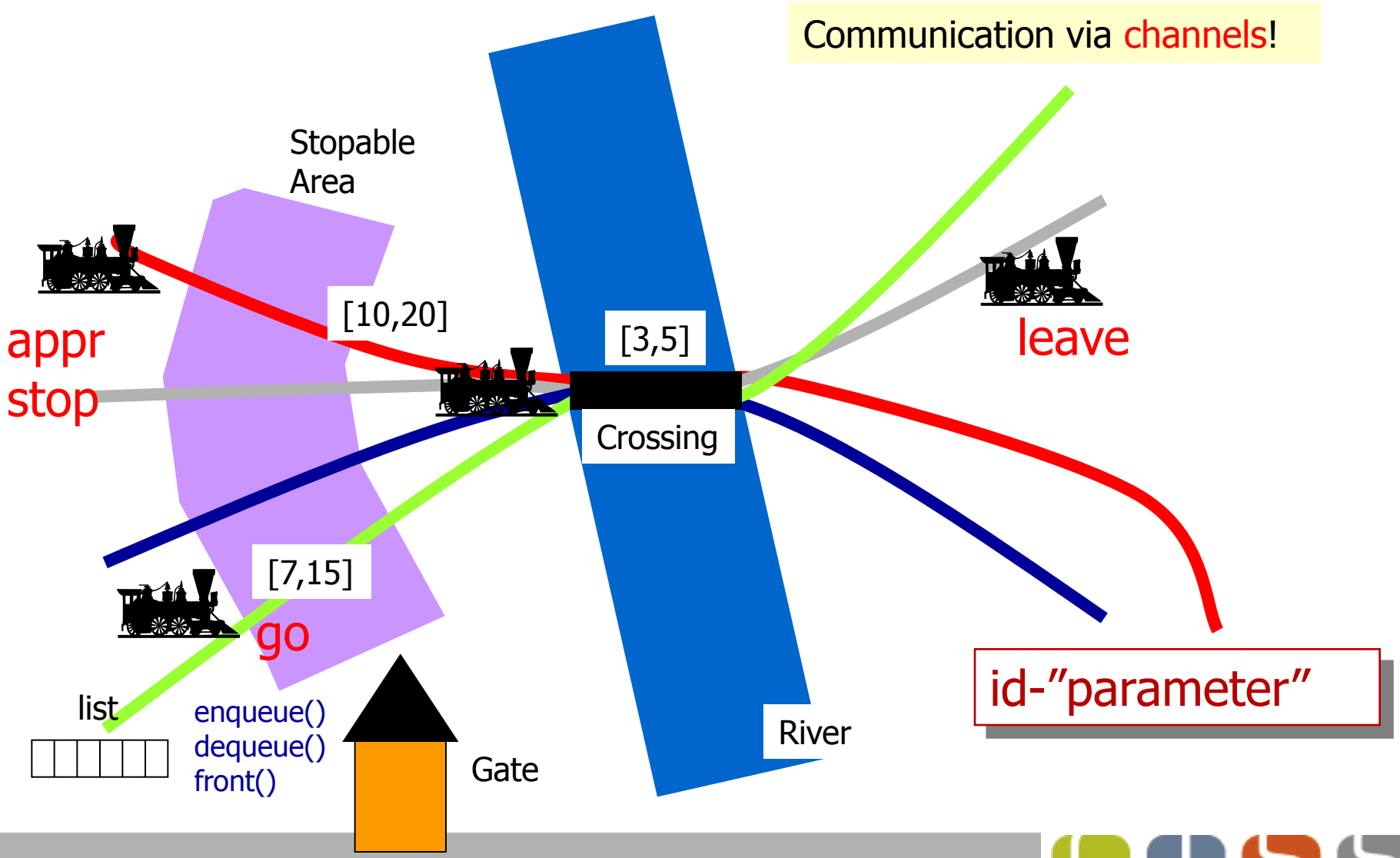
UPPAAL

Modeling & Specification

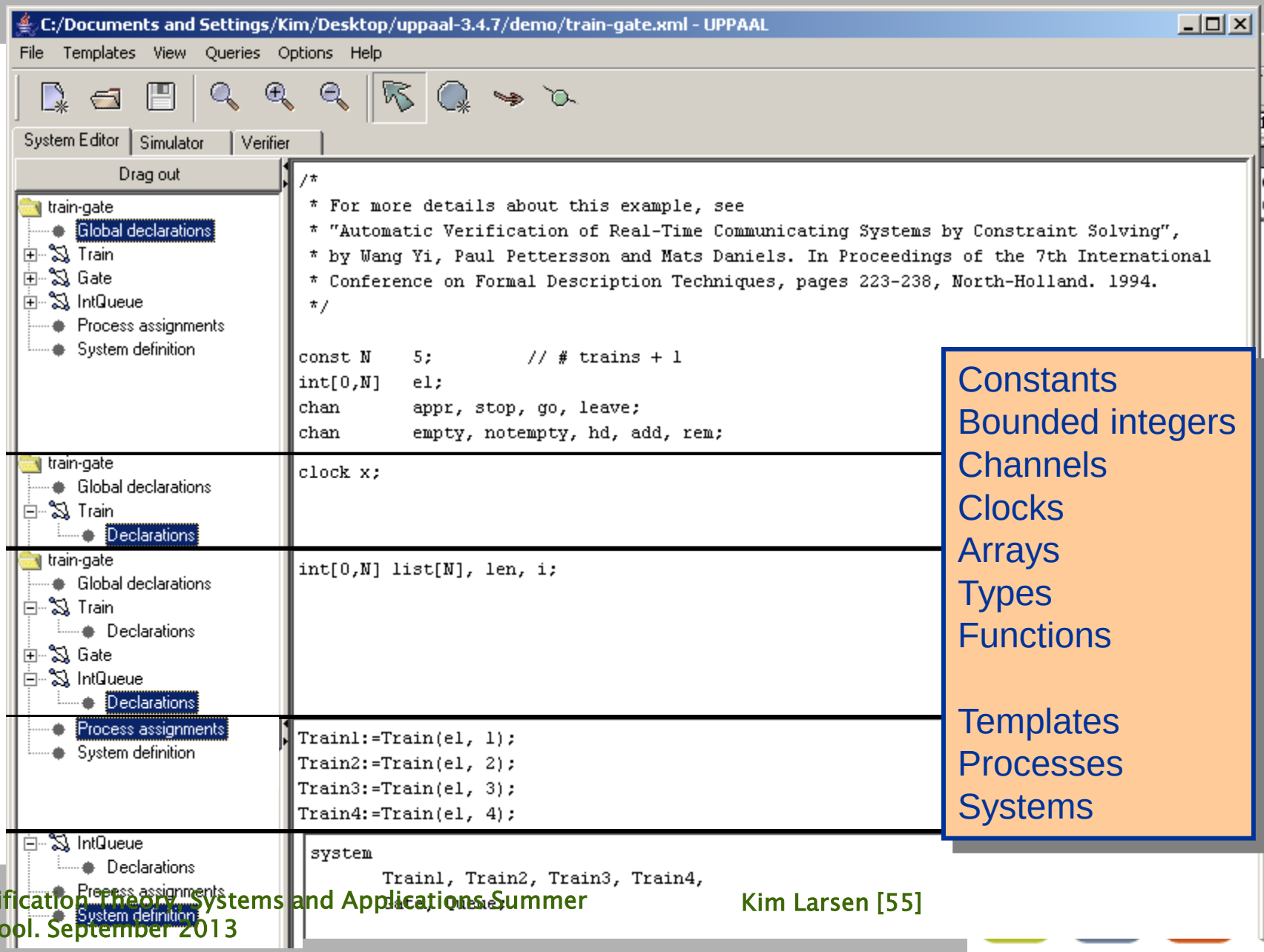
Train Crossing



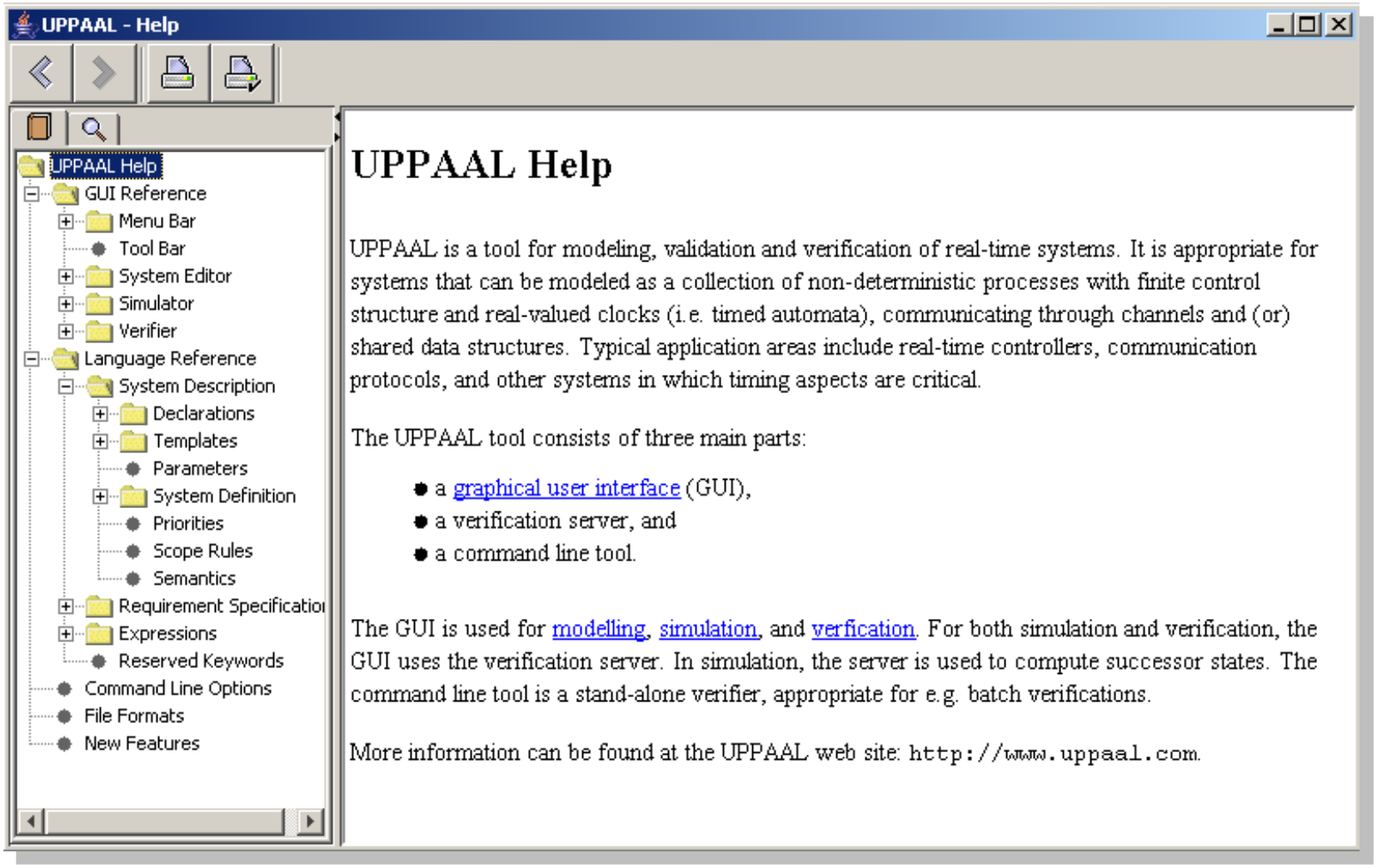
Train Crossing



Declarations



UPPAAL Help


UPPAAL - Help
UPPAAL Help
GUI Reference
Menu Bar
Tool Bar
System Editor
Simulator
Verifier
Language Reference
System Description
Declarations
Templates
Parameters
System Definition
Priorities
Scope Rules
Semantics
Requirement Specificatio
Expressions
Reserved Keywords
Command Line Options
File Formats
New Features
UPPAAL Help
UPPAAL is a tool for modeling, validation and verification of real-time systems. It is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks (i.e. timed automata), communicating through channels and (or) shared data structures. Typical application areas include real-time controllers, communication protocols, and other systems in which timing aspects are critical.
The UPPAAL tool consists of three main parts:
a graphical user interface (GUI),
a verification server, and
a command line tool.
The GUI is used for modelling, simulation, and verfication. For both simulation and verification, the GUI uses the verification server. In simulation, the server is used to compute successor states. The command line tool is a stand-alone verifier, appropriate for e.g. batch verifications.
More information can be found at the UPPAAL web site: http://www.uppaal.com.

Logical Specifications

- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{\leq t} Q$

The expressions *P* and *Q* must be type safe, side effect free, and evaluate to a boolean.

Only references to integer variables, constants, clocks, and locations are allowed (and arrays of these).

Case Studies: Controllers

- Gearbox Controller [TACAS'98]
- Bang & Olufsen Power Controller [RTPS'99,FTRTFT'2k]
- SIDMAR Steel Production Plant [RTCSA'99, DSVV'2k]
- Real-Time RCX Control-Programs [ECRTS'2k]
- Terma, Verification of Memory Management for Radar (2001)
- Scheduling Lacquer Production (2005)
- Memory Arbiter Synthesis and Verification for a Radar Memory Interface Card [NJC'05]

- Adapting the UPPAAL Model of a Distributed Lift System, 2007
- Analyzing a χ model of a turntable system using Spin, CADP and Uppaal, 2006
- **Designing, Modelling and Verifying a Container Terminal System Using UPPAAL, 2008**
- Model-based system analysis using Chi and Uppaal: An industrial case study, 2008
- Climate Controller for Pig Stables, 2008
- Optimal and Robust Controller for Hydralic Pump, 2009

Case Studies: Protocols

- Philips Audio Protocol [HS'95, CAV'95, RTSS'95, CAV'96]
- Bounded Retransmission Protocol [TACAS'97]
- Bang & Olufsen Audio/Video Protocol [RTSS'97]
- TDMA Protocol [PRFTS'97]
- Lip-Synchronization Protocol [FMICS'97]
- ATM ABR Protocol [CAV'99]
- ABB Fieldbus Protocol [ECRTS'2k]
- IEEE 1394 Firewire Root Contention (2000)
- Distributed Agreement Protocol [Formats05]
- Leader Election for Mobile Ad Hoc Networks [Charme05]

- Analysis of a protocol for dynamic configuration of IPv4 link local addresses using Uppaal, 2006
- Formalizing SHIM6, a Proposed Internet Standard in UPPAAL, 2007
- Verifying the distributed real-time network protocol RTnet using Uppaal, 2007
- **Analysis of the Zeroconf protocol using UPPAAL, 2009**
- Analysis of a Clock Synchronization Protocol for Wireless Sensor Networks, 2009
- **Model Checking the FlexRay Physical Layer Protocol, 2010**

Using UPPAAL as Back-end

- Vooduu: verification of object-oriented designs using Uppaal, 2004
- Moby/RT: A Tool for Specification and Verification of Real-Time Systems, 2000
- Formalising the ARTS MPSOC Model in UPPAAL, 2007

- Timed automata translator for Uppaal to PVS
- **Component-Based Design and Analysis of Embedded Systems with UPPAAL PORT, 2008**
- Verification of COMDES-II Systems Using UPPAAL with Model Transformation, 2008
- **METAMOC: Modular WCET Analysis Using UPPAAL, 2010.**

www.uppaal.org

RELATED SITES: TIMES | UPPAAL CORA | UPPAAL TRON

UPPAAL
Home

Home | About | Documentation | Download | Examples | Bugs

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).

The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University in Denmark.

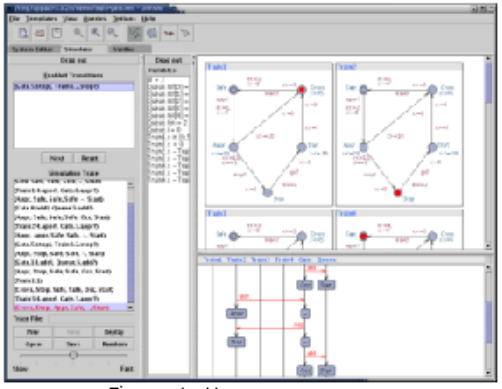


Figure 1: UPPAAL on screen.

Download

The current official release is UPPAAL 3.4.11 (Jun 23, 2005). A release of UPPAAL **3.6 alpha 3** (dec 20, 2005) is also available. For more information about UPPAAL version 3.4, we refer to this press release.

UPPSALA UNIVERSITET

AALBORG UNIVERSITY

License

The UPPAAL tool is **free** for non-profit applications. For information about commercial licenses, please email sales(at)uppaal(dot)com.

To find out more about UPPAAL, read this short introduction. Further information may be found at this web site in the pages About, Documentation, Download, and Examples.

Mailing Lists

UPPAAL has an open discussion forum group at Yahoo!Groups intended for users of the tool. To join or post to the forum, please refer to the information at the discussion forum page. Bugs should be reported using the bug tracking system. To email the development team directly, please use uppaal(at)list(dot)it(dot)uu(dot)se.

LAB-Exercises

Exercise 1 (Brick Sorter)

Excercise 19 (Train Crossing)

Exercise 2 (Coffee Machine)

Exercise 28 (Jobshop Scheduling)